VOLUME 07 | ISSUE 03A

digit Fast Track

YOUR HANDY GUIDE TO EVERYDAY TECHNOLOGY

to Careers in Gaming

You can make money building games?

So, what's a game?

Game development in India

Cellphone gaming

The people behind the games

The game development process

Tips – What works and what doesn't

A 9.9 Media Publication

www.
thinkdigit/forum

Join the forum to express your views and resolve your differences in a more civilised way.

thinkdigit
FORUM

Post your queries and get instant answers to all your technology related questions

One of the most active online technology forums not only in India but world-wide

JOIN NOW

www.thinkdigit.com

FAST TRACK to

CAREERS IN GAMING

powered by

CHAPTERS

CAREERS IN GAMING
MARCH 2012

CREDITS
The people behind this book

EDITORIAL
Executive Editor
Robert Sovereign-Smith

Writers
Arpita Kapoor & Mohit Rangaraju

Copy Editor
Infancia Cardozo

DESIGN
Sr. Creative Director
Jayan K Narayanan
Art Director
Anil VK
Associate Art Director
PC Anoop & Atul Deshmukh
Visualisers
Prasanth TR & Anil T
Contributing Designer
Vijay Padaya

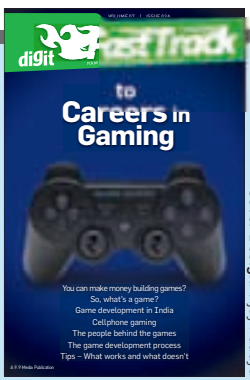
thinkdigit.com

March 2012

Free with Digit. If you have paid to buy this Fast Track from any source other than 9.9 Mediaworx Pvt. Ltd., please write to editor@thinkdigit.com with details

Custom publishing

If you want us to create a customised Fast Track for you in order to demystify technology for your community, employees or students contact editor@thinkdigit.com



Cover Design: Vijay Padaya

thinkdigit.com

Are you game?

Gaming, an industry which was considered to be dead in the early 80s, is now the most profitable industry among all entertainment businesses. With big players such as Microsoft, Sony and Nintendo entering the market, Gaming has grown to become a $67 billion industry and is expected to reach $112 billion by 2015. It will be bigger than Hollywood!

Even the Indian Gaming scenario has seen a vast change over the past 15 years. We now have an abundance of independent game studios trying their hands at making innovative games. With the count of MNCs going up in India, the job prospects have also risen and students no longer hesitate to choose Gaming as a career option.

With so much going on in the gaming industry, we're sure you want to know how it all works. That's what this Fast track tries to enlighten you about. There's hardly anyone who hasn't played a videogame in his entire life. In fact, some people spend their entire days and even nights playing video games. One such gaming enthusiast, Will Wright (the creator of *Sims*) claims that games were absorbing so much of his time, that he decided that maybe making games was the way to go. If you can connect to his story, then this is the right time to take the leap, seeing how well Gaming is doing right now.

We all love playing games, be it *Mario, Counter-Strike* or *Angry Birds*. Ever wondered how these games are developed? Is game development similar to software development? Making games requires a broad spectrum of skills such as programming, scripting, art, animation and music. Let's go into greater details of how the gaming industry can benefit and respond to everyone – right from investors and entrepreneurs to job seekers and gamers.

YOU CAN MAKE MONEY BUILDING GAMES?

You've probably played with the idea of making games for a living. You can, and reputed publishers don't have to back you

For years, independent game development was a hobby and revenue-generating games featuring alien creatures were created by large teams with loads of money in established game development companies. Before the mid-90s, commercial game distribution was controlled by big publishers and retailers and indie game developers were forced to either build their own publishing houses, or find publishers to distribute their games commercially. If none of that worked, all they could do was distribute it as shareware.

Only when e-commerce and digital distribution came into the picture did it become feasible for indie game developers to make money from their games. Digital distribution platforms such as Apple App Store, Chrome Web Store and Android Market and XBLIG (Xbox Live Indie Games) make it a piece of cake to distribute your game to a focused market.

With new platforms and numerous Open Source game engines and SDKs, it's now possible to make games with small teams and earn good money (if players love it, of course). Programmers can find lots of free art and graphics. Artists can either find freelance programmers to help them with the code or use starter kits or easy game engines such as Game Salad. Chapter 7 of this book discusses various tips and tricks which may help you in developing your own game.

To make a living from indie games, first get yourself a business-minded partner if the business aspect scares or bores you. If indie game development is not your cup of tea, you could master the required skills and try out your luck at game development companies as a programmer, artist or animator.

Before choosing Game Development as your career, you need to identify your interest and then build your skills accordingly. Whether it is concept art, UI design or background art? And what do you want to do – network programming, graphic programming or physics programming? Chapter 5 and 6 of this book will help you better understand the game development process and various roles that you could take up in a game company along with skills you need to develop to be eligible for those roles. Brace yourself – there's a lot to explore!

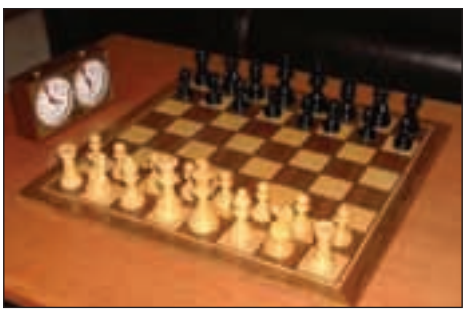
Next, let's look at what exactly makes a game a game.

SO, WHAT'S A GAME?

Quick, what's common between *Solitaire, Counter-Strike, Angry Birds* and *Mario*? That's right, they're all games; and despite their different target audiences, they share a set of common attributes

There are certain common elements that bind games in the genre of role-playing (RPGs such as *Diablo*), computers and console RPGs, massively multiplayer online games (MMORPGs such as *World of Warcraft*), puzzle adventures (*Limbo, Braid*), text adventures, action (*God of War*), first-person shooter (*Call of Duty* and *Crysis*) and casual mobile games (*Fruit Ninja* and *Angry Birds*):

- **Rules:** Rules clearly define possible and allowed actions of the players. Usually, the same rules apply to all players.
- **Interactivity and Player:** Games are player-centric and have lots of interactivity. They can be single player (Solitaire), player-versus-player (Chess), multiple players vs. the game system (World of Warcraft or FarmVille on Facebook.) etc.
- **Goals:** Every game has one or more goals without which interactivity is useless. Eliminating all opposing forces in *Counter-Strike* or killing pigs in *Angry Birds* are examples of goals.
- **Resources:** Resources can be anything under player control. These are given to the player at the start or during the game (maybe as rewards for playing better – some of which may be hidden at different levels.) The mushrooms and coins in *Mario* and various weapons in *Counter-Strike* are a few examples.
- **Struggle:** Struggle or conflict is introduced through opposition like enemies, non-player characters (NPCs) or puzzles depending upon the genre. The challenges involved make you play the game again and again.



Player-versus-player example: Chess

GAME DEVELOPMENT IN INDIA

How did it all begin? Who are the players in the industry? What are your options?

Inspired by a compelling demo of Shiny Entertainment's game MDK at an Intel Developer conference in early 1997, a young team of developers decided to get into game development. Intel was looking for companies willing to create software technology that was optimised for its upcoming Pentium2/AGP platform, and so they signed up with Intel for a project named "Project Dhruva". Since then, Dhruva Interactive has never looked back and is now completing 15 years in Indian Gaming. It develops games across multiple platforms and provides outsourced game content creation services to other countries.

Two years after Dhruva was started, IndiaGames – a five-member team – came about. Today it's a global operation with over 300 employees and offices in different countries including U.S, China and London.

Hungama Digital Media Entertainment started its journey the same year. It has to its credit Hungama Game Studio – which owns intellectual property rights to more than 400 Flash games till date for over 100 brands – and partnerships with DTH providers such as Tata Sky.

Two more players joined the fray after this. They were Paradox Studios (nowadays known as Jump Games) and Lakshya Digital.

EA Games came to India around this time and established its office in Hyderabad. With around 500 employees, EA Hyderabad has teams in three major global business units: EA Mobile, Central Development Services (CDS) and Global Finance.

In 2006, a number of India's leading videogame companies formed the Indian Games Industry and Trade Association (iGITA). It focused on creating an ecosystem for game development in India and fostering relationships between international and local game companies. The founding members of this association are Dhruva Interactive, Electronic Arts Asia, Hungama, IndiaGames, Mauj, Microsoft India (Entertainment & Devices Division), Mobile2Win, Paradox Studios, Small Device and Tinfo Mobile.

Games Club by Nazara Technologies

GameShastra, the developer of indig-

enous title *Desi Adda* started in 2006 and is now a global company with operations in India, US, UK and Japan. Around the same time, Trine Games, developer of *Streets of Mumbai* was founded as the game development and publishing division of Trine Entertainment Ltd., headquartered in Mumbai.

Then came Gameloft, which started mobile mass port and a quality assurance studio in Pune. In 2008, Ubisoft acquired Gameloft Pune and since then Ubisoft has released 100 per cent% Indian-made game titles for Nintendo DS, XBLA, iOS and Facebook.

Today brands such as Zapak, IndiaGames and Games2Win continue to be favorites on the web. And then you have names such as Zatun, BlueGiant Interactive, Kreeda Games, RZ2 Games and Robocule that exist alongside.

The new wave in the Indian gaming ecosystem

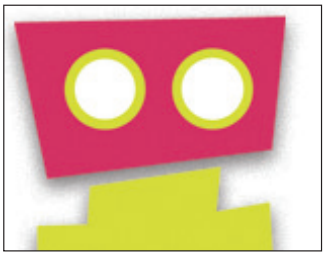
So much has happened over time. Today, we have a new breed of game makers called independent (a.k.a indie) game developers. In India, this includes Yellow Monkey Studios (*It's Just a thought* on App Store), MPowered.in (*Carmella* on App Store, Facebook, etc), IronCode Gaming, Over Cloud9 and Pyrodactyl Games. Studios such as Hashstash and Hungry Games are also working hard to bring out fresh games on various platforms.

Nazara Technologies has come up with a subscription-based model in the gaming sphere. GamesClub, developed by Nazara, is a first-of-its-kind inexpensive cross-platform gaming service, which offers unlimited games to the players for as less as ₹5 per day to just ₹99 per month. Playcaso, another Nazara property, vies to become a publisher of global repute with its co-production model.

Building communities

With organisations such as NASSCOM and FICCI taking interest in game development, the Indian game developer community has become more tightly bound. NASSCOM has successfully hosted three game developers conferences, and NASCOM GDC 2011 was a huge success. NASSCOM Gaming Forum helps developers come together on a common platform to share best practices and knowledge both, from the business and the development perspective. Gamers' Meetup (self-explanatory), an initiative by IndiaGames started in Mumbai and Pune with over 300 attendees meeting up. Indie GameDev India (InGDIn) is a group started to promote indie game development across India. The place enables programmers, VFX artists, writers, musicians, designers and others to collaborate. India

also has active local IGDA (International Game Developers Association) chapters in Pune and Bangalore. Also encouraging is the fact that various international contests including Dare to be Digital, Microsoft Imagine Cup, and Independent Game Festival Awards etc have been witnessing entries from India in recent years.



The InGDIn Community logo

FULL-TIME COURSES IN GAME DEVELOPMENT	
Institute	**Course/s offered**
DSK Supinfogame	Game Design & Production Management – 4 years
	Foundation Course – basics of Game Design and Development – 2 years
	Advanced course (for students with experience in industry) – 2 years
Backstage Pass, in collaboration with Jawaharlal Nehru Architecture & Fine Arts University	B.Tech (Computer Science & Game Development) – 4 years
	Bachelor of Fine Arts (Game Art & Animation) – 4 years
	Bachelor of Fine Arts (Game Design) – 4 years
National Institute of Design (NID)	Courses in Game and Toy Design
Image College of Arts, Animation and Technology (ICAT)	Courses in Game Design, 3D Game Animation and Game Development
Asian Institute of Gaming and Animation (AIGA)	Courses (degree and diploma) in Game Programming and Game Art and Animation
Srishti School of Art, Design and Technology	Course (advanced diploma) in Games Design
Kode Academy	Courses (diploma) in Game Development
Virtual Infocom	Courses in Game Design and Development
Sage School	Courses in Game Programming and 3D Design

CELLPHONE GAMING

Arcade consoles, personal computers, Nintendo consoles, playstations, Xbox... next what?

Smartphones and tablets are the best platforms to play games on due to their mobility and cost benefits. Thanks to games such as *Infinity Blade* and *GTA* available on tablets such as iPad, questioning graphic processing and memory capabilities of these new tablets is irrelevant. Gameloft has made seven of its existing titles available to consume the entire screen of the 7-inch Samsung Galaxy Tab. With smartphones such as Apple iPhone 4S, Samsung Galaxy and Android phones in the market, game developers have seriously considered them as gaming platforms as well. Also, awesome digital distribution platforms such as Apple App Store, Android Marketplace and Windows Phone Apphub have placed games just a click away from gamers.

Even with limitations of screen size and graphic processing, games on iOS and Android devices could generate more revenue than games on both, Sony's and Nintendo's portable consoles combined in 2011, according to a report by Flurry, a mobile analytics firm. However, there's no sign at present of smartphone games replacing consoles or mainstream gaming platforms since the applications of touch screens are limited, especially when it comes to core gaming. Consider *Fruit Ninja*, a game that requires just one control. Now compare it to a game such as *Call of Duty* that has separate commands for shooting, running, aiming, jumping, etc.

Nevertheless, thanks to these new age platforms, Indie developers have better chances of getting their games published. Let's have a quick look at some stats for the money made by these indie guys through mobile platforms. Rovio's *Angry Birds* was first developed for Apple's iOS in December 2009. Since

Gameloft Games on Samsung Tablet

then, over 12 million copies of the game have been purchased from Apple's Store. Simogo's *Bumpy Road* for iPhone and iPad has hit 100,000 downloads. This is a pretty big deal for the independent studio, which doesn't have the same name recognition as, say, a Firemint (developer of *Flight Control*) and is, obviously, competing against tons and tons of other independent studios in the ocean of games on the App Store.

THE PEOPLE BEHIND THE GAMES

Whatever game development be for you as an individual, overall it's a very complex process and needs a very broad range of skills. Let's find out what those skills are

The game industry today seeks individuals with focused specific skills and expertise though multi-skilled people always have a place. Every job in this industry requires you to have a passion for games. You should play games, the ones you like, the ones you hate and all the others in between. Passion and love for games is inarguably necessary. And then there are a list of other skills that fall under the "must have" and "should have" categories. If you understand what a job demands from you, you can prepare for that job from the start.

Often folks are confused game designers' job profiles with that of level designers', or designers' with artists'. In order to clear up all the confusion, we'll take a look at all the roles in an organisation. We'll delve into each role in detail for you, so that you also get a clear idea of the kind of job you would be more suited / qualified for, and thus be more focused with your applications to such companies. All jobs in game industry are basically divided into a few paths that you can specialise in:

- **Design:** This includes Game Designers, Level Designers, UI Designers, Systems Designers and at higher levels, also includes the Game Director, Creative Director, etc.
- **Programming:** This includes programmers skilled in the art of game programming, AI, graphics, networking and more
- **Game art and animation:** 2D/3D Animators, Concept Artists, UI Artists
- **Production:** Producers and Executive Producers
- **Quality Assurance:** Quality Testers and Software Engineers
- **Audio:** Audio often makes a game, and audio artists are essential to the process. Some talented souls here.

In this chapter, we'll look at each of these six varying roles in much greater detail to help you plan your way.

Design

Game development requires a lot of creative processes to happen and thus needs design expertise. Designers include Game Designers, Level Designers, UI Designers, Systems Designers, etc. Higher level positions include Game Directors, Creative Directors, etc.

Game Designers

"Working as a game designer is like the mouse on your computer; some days it clicks and some days it drags."

- Alan Emrich, Game Designer

Game development is a highly complex, intensive process, so the Game Designer must be able to work closely with teams of Programmers, Artists, Project Managers, Writers, Musicians and many others.

What does a Game Designer do?

- **Develop game concept, systems and content**
  Game designers collaborate with the Creative Director and the design team to develop core concepts, game systems, and game content. They are responsible for devising what a game consists of and how it plays and appeals to the player. They plan and define all the elements and components of a game, including its theme, architecture, rules, progression, set-up, story flow, characters, objects etc available to the characters; interface design; and game modes etc.

- **Design documentation**
  Game designers are required to write and maintain detailed game design documentation throughout the project duration. Game designers are required to write pitch documents, high-level design documents so that they can convey the game vision to all involved in the game process.

- **Conduct design reviews**
  Game Designers conduct periodic reviews of the game in progress and then balance and adjust game play experiences to ensure the product's critical and commercial success. They make adjustments to the original specification to respond to technical constraints and to be aligned with new programming/art developments in the game.

- **Manage design teams**
  Game Designers also manage the design team to successfully meet project objectives.

- **Workflow and scheduling**
  Game Designers coordinate workflow, scheduling and assignments with other team leads and ensures the information flow between both the designers and to other disciplines. It is the Game Designer's responsibility to ensure that the whole team understands the game concept and vision as outlined by the Creative Director. The Game Designer also trains testers to play the game, making sure that they understand what is expected of the finished product.

Must-have skills for a Game Designer

- Writing skills and professional documentation
- Thorough understanding of gameplay theory(What makes a fun game)
- Basic visual design and drawing skills (Paint, Basic knowledge of Photoshop etc)
- Designers must be good presenters and should have excellent communication /presentation skills.
- Storytelling and narrative development skills.
- Reasonably fluent in a range of 2D and 3D graphics and animation packages, such as 3D Studio Max or Maya.
- Good technical knowledge is required, with some programming skills at least at 'scripting' level.
- Knowledge of a game's target audience and market.
- Understanding of the capabilities and benefits of different hardware platforms (e.g., PC, console, mobile devices), as well as familiarity with software technologies and techniques appropriate to each platform.
- They must be able to work both as part of a team and independently.
- Originality, imagination and creativity.
- Systematic and strategic thinking.

Systems Designers

The Game Systems Designer is responsible for creating the game systems. He or she refines and maintains a variety of game systems.

What does a Systems Designer do?

- **Design Systems:** Create and design game systems covering a wide range of gameplay elements such as combat, user interface, levels, win/lose conditions and player objectives.
- **Define requirements:** Help define the requirements for the design of tools, pipelines, formulas and schemas.
- **Work with Production Team:** Work closely with members of the Production Team to execute design and work through design revisions as needed. They are responsible for aggressively prototyping and implementing new abilities, systems, and content.
- **Work with the programming team:** Define AI requirements for NPCs, mobs and bosses.
- **Review systems of the game:** Iteratively review, tweak and balance game systems.

- **Documentation:** Generate clear and understandable design documentation so that the documentation acts as a clear, stand-alone implementation guide to the entire team.
- **Presenting your ideas:** Present ideas in a concise manner, to audiences that are both technical and non-technical.

Must-have skills for a Systems Designer

- Experience with scripting languages such as Lua etc.
- Must be able to communicate well to their team.
- Strong mathematics background.
- Experience in pipeline and tools design.
- Proficient with Excel and Vision.
- Passion for implementing core game mechanics and game systems.
- Strong communication and technical writing skills.

Level Designers

Level Designer ensures that the game's levels deliver an unforgettable interactive experience for the player. Deciding the level flow, worlds, difficulty levels and variation among different levels of the game is the responsibility of the level designer. He understands the game vision, concept and mechanic to design interactive, balanced and fun levels for the game.

A level designer is part game designer, part 3D artist, part programmer, part architect.
-Tom Sloper, Sloperama Productions

What does a Level Designer do?

- **Create game environments:** Level Designers use design tools to create 2D/3D environments.
- **Design level gameplay:** The Level Designer develops the gameplay for the level, defines challenges that the player will face and allowed actions for the player. The Level Designer also implements game mechanics using different objects, places to hide, skills, weapons etc.
- Leverage proprietary tools to ensure all levels perform well on the target platform.
- Work closely with the Creative Director to understand, communicate and implement major game play systems and features into all levels.
- Review level difficulty and balance.
- Participate and host creative sessions, development meetings, reviews, and planning sessions.

- Clearly communicate to project leaders on current level layouts and status of level development.
- Create and maintain industry-standard design documentation, including specifications for level overviews, world building metrics, storyboards and flow charts for all or portions of levels, etc.
- Discussing with the programmers and artists to lay down a detailed inventory of level 'assets'

Must-have skills for a Level Designer

- Solid understanding of Physics, Geometry, Geology, and Classical Art.
- Working knowledge of design systems in order to build levels in 3D space, place enemies and completely understand the relationship between gameplay elements within the level and the overall game.
- Experience with level editors such as Unreal Editor, Radiant, Hammer, Scrolling Game Development Kit etc.
- Experience on art software such as 3D Studio Max, Maya or any other 3D world building software, Adobe Photoshop etc.
- Knowledge of Scripting Languages such as Lua , Python , Unreal etc
- Level and play-flow planning.
- Excellent writing and communication skills.
- Knowledge of level optimisation techniques such as portailing.
- BSP block-out
- Knowledge of detail geometry and lighting.
- Terrain creation and sculpting, unwrapping and normal map creation
- Mutli-player and single-player level creation.
- Modding' experience by making mods of already published games.

User Interface / User Experience Designers

"A user interface is well-designed when the program behaves exactly how the user thought it would."

- Joel Spolsky

Players/users are the most important aspect of a game – after all they are the ones who play your game. As a UI/UX Designer, you're responsible for synthesising user experience requirements from game designers, and creating the most elegant UI system to support the needs of players to make it a smooth ride for them to locate options, resources, etc.. UI designers work closely with the art team to take required assets for implementing the UI.

What does a UI/UX Designer do?

- Design logical information architecture and detailed wireframes.
- Define and lead UI prototyping.
- Understand and accept feedback from the usability research department and make necessary changes.
- Work closely with the design team to determine the functionality and features of the user interface.
- Work with the art team to concept the user interface, to ensure a consistent art style is maintained across the game.
- Work with the programming team to ensure the UI implementation matches the game-play intention.
- Use game engine tools to integrate UI assets and functionality into the game.

Must-have skills for a UI/UX Designer

- Deep understanding of UI design theory for games and best principles of interactive design.
- Understanding of usability of interface of software especially games.
- Strong skills in crafting effective page flows, layouts, and wireframes to convert technical documents to UI flow diagrams.
- Knowledge of using game tools or scripting languages for tweaking UI elements and enhancing UI effects.
- Fluent with Adobe CS5 products and common design tools.
- Skills in merging game frontend layout design with overall game concept.
- Ability to work with several development teams, while maintaining consistency of design.
- Good writing and communication skills.
- Understanding of the role of typography in UI/UX design.

Game Writers

Game writers simply help designers craft an immersive, interactive narrative experience which may/may not include dialogs, cut scenes etc. The role of a game writer is to simply start the game at an interesting point, climb its way over a few emotional peaks, and end it even more interesting.

"Story in a game is like a story in an A-rated movie. It's expected to be there, but it's not that important."

- John Carmack

However, attitudes are changing now. From mainstream console titles to iPhone games, almost all games need a story or a narrative and have a clear set of emotional shifts, tonal changes, and meaningful moment-to-moment events that generate emotional pay-offs.

What does a Game Writer do?

- Create concepts, story elements, and pitch documents for mainstream game story and key story moments working closely with game design team.
- Write and revise game dialogue as declared by gameplay requirements.
- Accept feedback from fellow writers, design team and development team.
- Write and revise cinematic scripts for cut-scenes, trailers etc.
- Create and maintain NPC character information, world back story, and scope for sequels and define emotional experiences of the game.
- Collaborate with art team to effectively display the character traits etc defined by your story.

Must-have skills for a Game Writer?

- Good writing skills including ability to write interesting dialogues, narratives etc.
- Flexible in conducting re-writes and changes in game story as needed by design and development team.
- Ability to work collaboratively with a team of writers and with a development team and ability to accept feedback and make required changes.
- Understand techniques to ignite player emotions.
- Strong research skills and extraordinary attention to detail.
- Excellent knowledge of writing tools, solutions for content creation such as MS Word, MS Excel, database solutions and script formatting tools.
- In-depth knowledge of industry trends in interactive storytelling and various emerging story-telling trends like parallel story-telling, multiple-ending stories etc.

Programming

"The concept for PITFALL took less than 10 minutes. The difficult part was sitting at the computer, for over 1,000 hours, and making it happen."
- David Crane

Video Game Engineering is an intellectually demanding and challenging

job. The programming problems involve physics ranging from collision detection, gliding movements, forces to complex physical transformations, from making more realistic characters using AI techniques to path finding, from advanced rendering techniques to optimisations including memory needs and size/platform issues.

Game Programmer

"A game programmer is someone who actually implements the game on the computer, turning the ideas and designs into an actual program. Despite often having to work long hours and rarely getting their photographs in games magazines it is a fun job and can be very rewarding."

- Marc Vaughan

However, as you're the one who actually implements each and every feature of the game, it's on you to give an experience to the player that a designer imagined, taking it from a mere idea in theory to a game .

What does a Game Programmer do?

- Working with existing game engine code to implement various gameplay, AI, mechanics and rules of the game.
- In case of a fresh product, may be you need to write the core engine from the scratch.
- Debugging and optimising console and PC performance.
- Specifying testing procedures, taking feedbacks and making requisite changes in the code.
- Managing a team of game programmers to create immersive and engaging game play experiences in case of larger teams if in a role of Lead Game Programmer.
- Developing well designed software within project schedule to team standards and schedule.
- Responsibility of documenting code features.
- Safeguarding of the company's assets including source code, art work, tools, In-House engines etc.
- Defining game frameworks and architecture.
- Keeping up to date with the market, scan new development tools and try innovative technologies.

Must-have skills for a Game Programmer

- Experience with build tools (SVN, Maven etc).

- Excellent cross-platform, multi-language programming skills (or choose a platform/language you're comfortable in)
- Strong technical background preferably a computer science/IT degree.
- Expertise in at least one programming track (graphics, animation, AI, sound etc)
- Experience in translating game design into technically feasible goals.
- Ability to adapt to quickly changing project requirements and iterative feedback.
- Knowledge of modern game content production process/ tools/ engines.
- Strong 3D and mathematics skills, sharp logic and strong problem solving skills..
- Familiarity with bug-tracking systems.
- A passion to make and play games.
- Demonstrated knowledge of good software engineering practices.
- Good communication and writing skills.
- Demonstrate continuous learning and growth.

Artificial Intelligence Programmer

An Artificial Intelligence (AI) Programmer plays a key role in making realistic, believable characters, environments, etc., in computer games. An AI Programmer writes code to simulate real world behavior and intelligence in computer enemies and opponents. Next-gen console and PC games have intelligent path finding systems and it is becoming a challenging task to develop more and more sophisticated AI in computer games. Games may need only few AI Programmers or may need many of them depending upon the need of the game. Real-time strategy games, Role-laying games such as *Civilization III* use AI heavily while others such as *Limbo* or puzzle games use lesser or almost no AI implementation.

For example, the game *Black & White*, developed by Lionhead Studios features a unique AI approach to a user/player controlled creature that uses learning to model it's behaviors during the course of the game.

What does an AI programmer do?

- Design and implement advanced and realistic AI to create an immersive game experience for the player.
- Implement AI routines that control the intelligence and movement of characters, vehicles and world objects.
- Research and keep yourself updated with new cutting-edge AI techniques.

- Document technical design specs in technical design document.
- Design, maintain, implement, test and debug AI code.
- Development and maintenance of reusable advanced AI tools, features and pipelines.
- Find and fix AI bugs flagged by the Quality Control team.

Must-have skills for an AI Programmer

- Demonstrated knowledge of good software engineering practices.
- Understanding of memory management, multiple processor use, and runtime optimization.
- Strong C/C++ programming skills
- Strong Object Oriented design skills
- Strong 3D Math, Trigonometry, Calculus and Linear Algebra skills
- Experience in self/college project: Path finding, locomotion, collision detection, finite state machines, game logic etc
- Experienced in implementing character AI systems in real-time games.
- Knowledge of Neural Networks, Genetic Algorithms, Fuzzy State Logic, and A-Life.

Game Tools Programmer

"Right now we have talent and tools, but soon we'll have talented tools. New intelligent software will help game developers speed up the process as never before."
-Duane Alan Hahn (2005)

The tools/engine programmer writes tools that can handle game-specific tasks. These tools may/may not be included in the game but help the game development process accelerate. A bad tool can hamper the game dev process where as a good tool can make it a piece of cake. These tools include scripting tools, model viewer, map editors, level editors, dynamic map generators, and custom tools for a specific game. They help the artists and game designers interface with the game core engine.

What does a Tools Programmer do?

- Program tools that help designers, artists etc to make their work easier.
- Document the tool feature to help artists and programmer utilize them fully.
- Participate in the design and implementation of games.
- Collaborate with Producers, Designers, third parties and other technical team members to plan and implement tools and pipelines.

- Assist in troubleshooting and resolving tools / pipeline related issues.
- Support game teams in feature development.

Must-have skills for a Tools Programmer

- Excellent C# understanding and skills. The ability to write highly efficient and optimized code.
- Strong understanding of the game engine.
- Knowledge of the target systems (PC, or one of the consoles).
- Solid understanding of C/C++, possibly Assembly, mathematical concepts, graphics, collision detection, object oriented programming, and database management.
- Good communication skills so as to understand the requirement of the artists and designers
- Knowledge of user interface design as a tool is great if it has a handy interface.
- Clear and neat documentation skills.
- Excellent debugging skills.

Graphic Programmer

A Graphics Programmer in particular must master the full suite of techniques to realise three dimensional objects inside a two dimensional display and must develop programs to handle complex 2D and 3D graphics.

What does a Graphic Programmer do?

- Develop core graphic engine and rendering systems
- Designing and implementing systems and tools to support rendering and technology needs (such as lighting, occlusion, shaders).
- Implement realistic and fascinating particle effects.
- Implement/improve visual effects using current in house technology.
- Collaborate with Art Direction and Design to find the best ways of generating great graphics with minimal performance costs.

Must-have skills for a Graphic Programmer

- Math expertise (especially linear algebra and advanced calculus).
- Expert in modern rendering and engine-related issues.
- Knowledge of writing shaders.
- Excellent code/performance optimization skills.
- Knowledge of graphic APIs

- Extensive knowledge of databases and operating systems.
- Expertise in multithreaded programming
- Strong understanding of software development practices
- Knowledge of tools such as 3DSMax, Photoshop, Flash.
- Strong C/C++ language skills
- Good knowledge of C# language
- Strong artistic sense.
- Knowledge of HLSL, GLSL etc.

Network Programmer

A Network Programmer writes code for implementing the game over a network to allow multi-player mode in computer games. This allows players to compete or cooperate together in the game whenever they are connected via LAN, Wireless, and Bluetooth etc. This is one of the technically most challenging jobs as the programmer needs to have strong knowledge of networks along with basic understanding of code implementation. Network Programmers are in great demand due to popularity of online games.

What does a Network Programmer do?

- Implement a complete game server.
- Integrate and implement core online components on multiple hardware platforms.
- Work closely with gameplay programmers to precisely replicate game states between client and server side.
- Handle latency and synchronization issues.

Must-have skills for a Network Programmer

- Understand standard programming fields: client/server architecture, network security, basic network protocols (e.g. TCP/IP or UDP), concurrency, multi-threaded code, synchronization, and network APIs like DirectPlay and Winsock.
- Must have a deep understanding of Database creation, management, and administration in case of online multi-player games
- Knowledge of programming in C/C++/C#.
- Familiarity with game service APIs (PSN, Xbox Live, Windows Live).
- Knowledge of how to build robust, scalable backend systems and tuning those systems for performance and reliability.

Other programming roles

There are certain programming roles which usually are not a job title, or even a full-time position on a particular game project, but are still important tasks in their own way These include Input Programmers who take care of implementing input systems for the game, sound programmers who implement game sounds properly in the game and UI programmers who take care of scripting UI elements of the game.

These need basic programming skills and understanding of game systems and content pipeline. For real-time motion-controlled game utilising devices such as Kinect, we need expert input programmers who can handle complex input methods and maintain low input latency.

Art and Animation

Art and Animation are as vital to games as to animation films and a cartoon TV series. Games need Artists, Animators, Visual Effects Specialists, etc., to give the game the exact look and feel as perceived by the Game Designer and work closely with the development team to keep providing them art assets when needed.

Various expertise/focus roles one can take-up as an artist/animator are:

Technical Artist

The Technical Artist (TA) acts as a link between the Artists and Programmers working on a game project. They ensure art assets can be easily integrated into a game without sacrificing the overall look and feel of the game as presumed by the designer keeping in mind the technical limits of the chosen platform.

What does a Technical Artist do?

- Work closely with both the art and the programming team to make sure that the art assets reflect and integrate successfully in the game.
- Decide and maintain the art production workflow.
- Decide on the choice of art tools to be used.
- In a more of an advisory position, set up the systems of production as well as solve problems as they arise.
- Create custom tools to improve the efficiency of their team using the scripting languages included in major modeling packages.
- Provide feedback or debug complex assets such as skinning systems etc.

Must-have skills for a Technical Artist

- Updated knowledge of changes in technology, both in terms of console hardware, art packages, and new techniques.
- Strong knowledge of both art and programming.
- Detailed knowledge of at least one industry-standard art packages like Abode, Autodesk etc.
- Knowledge of scripting languages
- Skills in lighting, modeling, rendering and knowledge of shaders.

Specific tasks in more focused roles:

1. **Shader Writer:** They create efficient real-time shaders using HLSL, GLSL etc. They need strong mathematical background and an artistic view.
2. **Effects:** They take care of design and implementation of realistic in-game effects. They create tools and systems to implement particle effects, hair, cloth, fur and even lighting and compositing.

2D/3D Animator

Animators are responsible for creating movement and behavior of game characters, objects that populate game world and environment. Animators make 2D/3D animations to make the game world and characters look realistic and believable.

What does an Animator do?

- Make animated objects/vegetations/action scenes etc using software packages for the game.
- Make smooth, believable character animation to support gameplay.
- Make cinematic animations including multiple cut scenes, background animations etc.

Must-have skills for an Animator

- 3D/2D animation skills using Maya, Flash (or any other 3d/2d animation package)
- Knowledge of Premiere/After Effects
- Understanding of workflow, tools, motion capture, rigging, character skinning, scripting, cameras, cinematic animation and setup, and
- Knowledge of engine parameters/compression/optimizations etc.

Concept Artist

Concept artist is primarily involved in the creation and design of characters,

structures and worlds found within video games. Designs must fit the theme of the game and colors must be considered and utilized, to form a balanced scheme throughout the design of the game. A concept artist has to come up with new innovative characters, objects that populate the game world.

What does a Concept Artist do?

- Creates and designs the structures, characters and objects that conform to the vision of game designers.
- Is responsible for a consistent look and feel of the game.
- Research real world designs to get inspired from them.

Must-have skills for a Concept Artist

- Understand the theory of composition and color.
- Rendering exciting industrial design.
- Grasp of the practical aspects of content creation.
- Proficient life sketches/drawings abilities
- Good observation skills
- A good grasp of anatomy proportion and skeletal structures in life drawings.
- Knowledge of Illustrator, Photoshop etc.

Environment Artist/Modeler

An Environmental Artist/Modeler is the one who understands the game-designer's vision of the game-world and builds the 2D and 3D worlds and components, especially organic and inorganic environments. It's usually the Environmental artist's work that makes a player feel "WOW" after looking at the game. They do a lot of work in 3D art packages, custom level layout tools, and in texture artwork packages like Photoshop.

What does an Environmental Artist do?

- Creates 2D and 3D worlds with the help of 3D and 2D artwork packages.
- Environmental artists work with level-designers to create architectural designs, themes and compositions
- Environmental artists work very closely with the programmer to generate new and innovative looks, lighting and moods in their levels.

Must-have skills for an Environmental Artist

- Have an ability to think in 3D.

- Have a sense of form, weight (mass), volume, scale and level of finish.
- Adept at modeling, shading and texturing in any relevant 3D package
- Have solid knowledge of the technical aspects of level building, and have constructed levels from start to completion.
- Technically sound to make the programmer understand what you have.
- Good communication skills to interact with Level-Designer, Game-Designer and programmer.
- Have a good knowledge of how to use lighting and shadowing techniques.
- Good drawing, coloring and texturing skills.

UI Artist

UI artists are the ones who think about player's experience and interaction with the game. Their goal is to make the player's experience as soothing as possible and reduce ambiguity. Their focus is to see that the player can easily navigate through the menus, icons and HUDs (Heads up displays).

What does a UI Artist do?

- Conceptualize, Design and create individual 2D assets for the interface.
- Animate the icons and menus if required.
- Creates 2D/3D assets
- Competently assists in design of UI flow and function
- Works with Tech. Programmers and Game Designers
- Develop common interface ergonomics like creation of broadcast style overlays for in-game presentation and functionality of the products.

Must-have skills for a UI Artist

- Have strong artistic, typographic and layout skills
- Should have the knack of creating innovative design solutions.
- Thorough understanding of digital/traditional and Graphic art techniques
- Thorough understanding of UI flow and function design
- Good verbal and written communication skills and ability to interact effectively.

VFX Artist

The VFX Artist is responsible for the special visual effects that appear in a game. They are the ones who make natural effects like water, smoke, foliage, dust, fabric as well as fantastic effects like explosions which makes the game appear more real.

What does a VFX Artist do?

- Conceptualize, design and create visual effects in games.
- Create dynamic VFX animations and apply compositing techniques to a variety of VFX assets.

Must-have skills for a VFX Artist

- Proficient in Photoshop and After effects.
- Proficient in high-end CG effects systems utilizing particle systems, dynamics and procedural animation (e.g. Maya, 3Ds Max).
- Proficient in scripting languages (Mel, Maxscript or Python).
- Solid Technical abilities to aid in creating new visual effects art pipelines.

Character Modeler

What does a Character Modeler do?

- Modeling and texturing high quality character models in low, medium and high polygon counts.
- Rig models of characters before delivery to the animation team.
- Partnering with animators to ensure character skeletons and rigging are appropriate.

Must-have skills for a Character Modeler

- Strong Digital painting and texturing skills.
- Clean low-polygon, organic modeling skills.
- Strong proficiency in 3DS Max, Photoshop
- Understanding of efficient models and texture layouts.
- Strong drawing skills.

Texture Artist

These are responsible for painting the walls of buildings, surfaces of environments, and objects in the game. Texture Artists work very closely with Environmental Artists and help to increase the speed at which worlds come to life. Sometimes, they use real-life photographs as a basis for a texture and then use their skills to edit or otherwise manipulate the imagery.

What does a Texture Artist do?

- Works with vector and raster image creating tools for creation of all textured surfaces.

- Create high quality texture maps, including Bump, and spectacular Maps, diffuse, normal, ambient occlusion, reflection, and environmental maps.

Must-have skills for a Texture Artist

- An ability to create complex textures while using minimal necessary colors, resolution.
- Proficient in computer graphics lighting.
- Proficient in Vector and raster image creation tools
- Knowledge of appropriate use of light/shadow and colour theory.
- Knowledge of how palettes and CLUTs (Color look up table) work.

Production

Project Managers play an important role – be it games or any other field. In games, Producers are responsible for timely completion of projects. This term was introduced by Trip Hawkins, founder of Electronic Arts. Producers are the ones who manage artists and developers.

Producer

What does a Producer do?

- Contact and Negotiate contracts, including license deals.
- Act as liaisons between development staff and upper stake-holders.
- Manage Art, Programming and Design teams.
- Scheduling the project.
- Supervise project design and development.
- Draft deal points as the basis for contracts prepared by legal/business affairs.
- Set goals and make the team achieve them.
- Direct the other product elements such as Marketing, legal, public relations and operations activities.

Must-have skills for a Producer

- An MBA or a law degree I beneficial.
- Knowledge of market trends.
- Effective use of hardware and software interfaces.
- Good communication and interpersonal skills.
- Proficient in project scheduling and managing.
- Strong Technical background is a plus.

Executive Producer

The Executive Producer (EP) is equivalent to a combined Director and Producer from traditional media. EP is both creative visionary for the product and the person who builds the team and motivates them to accomplish their goals. It is not an entry-level position and requires an experience as a producer for at least 4-5 years.

What does an Executive Producer do?

- Develops creative center of the game.
- Builds a playable/shippable level demonstrating the key attributes of the game.
- Communicate the core experience of the game to development teams.
- Works with key reports (Development Director, Art Director, Audio Director, Creative Director and Producers) to create an organizational structure and processes that allow the team to efficiently produce high quality work on time and on budget.
- Articulate goals and team principles properly to everyone in the team.
- Sell the value of the game across industry, interviews and meetings with press.

The skills are similar to the producer but you need stronger managerial skills.

Quality Assurance

The people in QA department assure the quality of games being delivered in an organization. They play test the games and are in charge of fining bugs, if any. This is the dream job of many hard-core gamers, as you get paid to play games. People involved in this department:

QA Tester

Testers play a game over and over again, testing different versions of the game. People initially feel that it's fun; but later find it to be tedious and repetitive. Testing is generally done in teams. The testers have limited influence over the design. This is a proper entry level role.

What does a QA Tester do?

- Play the game and find bugs in the game.
- Categorise and prioritise the bugs and properly document them.
- Communicate the bugs to the Designers.
- Work on deadlines.

- Understand the Production and marketing schedules.

Must-have skills for a QA Tester

- Be a passionate gamer.
- Some programming knowledge is a plus.
- Good communication and documentation skills.
- Understanding of how games are put together, and how the different elements of a game contribute to the playing experience

QA Software Engineer

What does a QA Software Engineer do?

- Provide solutions to improve quality of games.
- Provide solutions for debugging; build verification, automated testing, and data analysis.
- Enable requirement based technique, unit testing and integration testing by implementing proprietary and commercial tools.
- Work closely with test leads and development leads to continually evaluate and improve the software development and testing process.
- Develop tools and systems in support of testing or debugging on schedule and specification.
- Follow best coding practices.
- Document all the works done.

Must-have skills for a QA Software Engineer

- Competence with some of the current development technologies an platforms including C#, C++, Java, PHP, SQL.
- Ability to multi-task, prioritize and be flexible
- Good verbal and inter-personal skills.
- Technical writing skills.

Audio

Audio plays an extremely important role in games. A suitable audio makes a player more involved. People involved:

Audio Artist

What does an Audio Artist do?

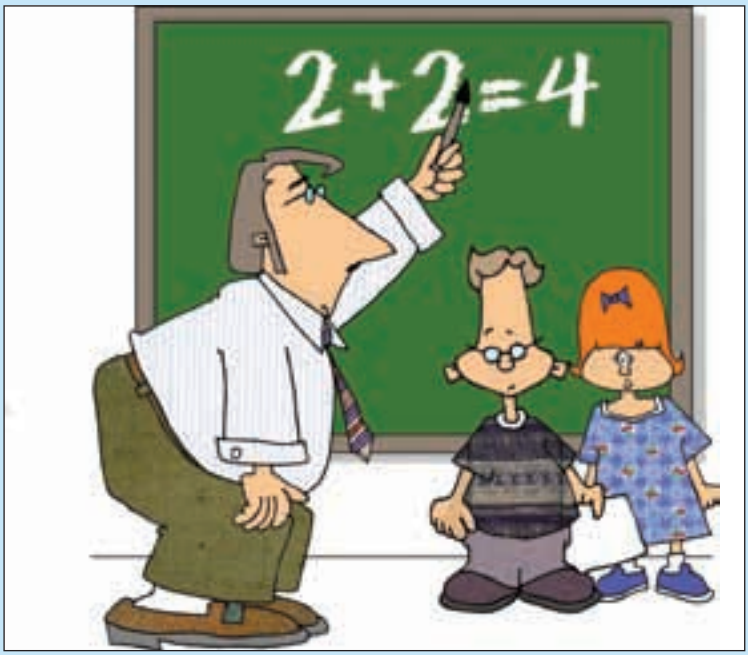
- Creates a game's sound-track, sound effects, character's voices, spoken instructions and ambient effects.

- Produce a sound design for the game.
- Designing, creating, editing, and mixing of some original audio content in accordance with game project's design goals.
- In-game implementation of all types of audio assets using proprietary tools and middleware solutions

Must-have skills for an Audio Artist

- Exceptional sound design skills and strong technical audio vision
- Experience creating audio pipelines on projects
- Ability to make high quality audio samples and process externally created music, VO, and SFX for in-game use.
- High degree of proficiency using professional audio software and hardware.
- Dialogue recording experience is a plus.
- Ability to write/debug simple computer script or batch files

Thus, we see that game development process needs expertise from a variety of fields and all these experts and teams work hard to get those great games to your hands to play.



THE GAME DEVELOPMENT PROCESS

Now it's time to see what stages you will go through when developing a game

To release any final product, one needs to follow a process, and game development is no exception, and there are three distinct phases in the process. These are very similar to the ones followed in software development life cycles (SDLC). There's this waterfall model, which is rarely followed. The most common model followed is the Iterative with Rapid Prototype model. In the Iterative Design with Rapid Prototype model, designers focus on game-play over graphics. Many game designers prefer paper prototyping of video games as it is easier to test iteratively. Once the prototyping is done, play-test takes place. This might be a full or partial play session to identify strengths and weaknesses in designs. Then, corresponding changes are made to the design depending upon the reviews after the play-test and this process is repeated iteratively.

The game development process is broken down into four phases:

1. Pre-Production
2. Production
3. Post-Production
4. The Aftermath

In this chapter, we will see how a game is developed and what activities are conducted during these phases so as to get a clear and solid understanding of game development process.

1. Pre-Production

This is also known as the design phase. In this phase, the main outline of the game, planning, scheduling and documentation is done. This phase is very important to pitch a game to the publishers. This phase is not funded by publishers.

Before understanding the design process, one needs to know what game-play and game-mechanics mean. These are known as the atomic elements of game design.

Players, avatars and game bits

In majority of the digital and non-digital games, players are represented either by an avatar or game-bits like many RPGs, FPS, Monopoly. In some of the games, the player has no avatar and the player represents himself or herself. This is true in case of games such as *Poker* or *Risk*; where little soldiers and artillery pieces are defending the player – the unseen leader. It is very important in a game to define this part.

Game Aesthetics

The MDA (Mechanics-Dynamics-Aesthetics) model describes the three important elements of a game. The first one we'll be discussing is the Game Aesthetics. Game Aesthetics describes the emotional responses evoked in the player, when the player interacts with the game system. Aesthetic in short is the feeling that you want the player to have after/while playing the game. It describes the play experience of the player. The MDA model classified eight aesthetics in general that are seen in games and example games they can be observed in:

- Sensation: *Quake*
- Fantasy: *Final Fantasy, Sims, Quake, World of Warcraft*
- Narrative: *Final Fantasy, Sims, Quake*
- Challenge: *Charades, Quake, Final Fantasy*
- Fellowship: *Charades, World of Warcraft*
- Discovery: *L.A. Noire, Sims, Final Fantasy*
- Expression: *Charades, Sims, Final Fantasy*
- Submission: *Final Fantasy*

Game Mechanics

What really makes the game space a very interesting place to be are game mechanics."A game mechanic" is another term for what others might commonly call a "rule." Among those in the industry, though, the term "mechanic" is commonplace. Mechanics are how something works. If you do X, then Y happens. If X is true, then you can do Y. In Monopoly, if you land on a property, you can buy it. If you roll the higher number, you get to go first. Each is a simple mechanic.

Designing Mechanics is as fun as cooking. Amazingly fun things can be made with the help of Game Mechanics. Mechanics are the rules of the game that act upon the players, avatars and game bits, game state and game views, and describe all of the ways to change the game state. Mechanics are in a way ingredients of Game Design.

Here are some common classes of mechanics that you can put in your game:

- **Setup:** There must always be at least one rule that describes how the game begins.
- **Victory conditions:** There must always be at least one rule that describes how the game is won. Some games, like open-ended role-playing games (RPGs), have no victory condition. As a result, some designers do not

consider them games. Others consider the achievement of a specific goal a victory as they race off to the next goal.

- **Progression of play:** In Board-games, the things to be considered are: Who goes first, and how? Is the game turn based or real time? For turn-based games, does the game start with one player and then proceed clockwise, or do players bid resources in an auction for the right to go first each round, or is there some other method? For real-time games, when two players try to do something at the same time, how is that resolved? In many video-games, the progression is real-time, where actions are taken as fast as players can take them. Hence, here sequencing is important. Questions like: "What will the player do next? What will keep the player motivated to go forward?" are answered with the help of Progression Game Mechanics.
- **Player actions:** Sometimes referred to as "verbs," some of the most important mechanics describe what players can do and how will it affect the game state.
- **Definition of game view(s):** Mechanics define exactly what information each player knows at any given time. Note that some mechanics may change the view, such as partially lifting fog of war in an RTS under certain conditions.

Game Dynamics

According to the MDA model - "Game Dynamic is defined as the pattern of play that comes from the mechanics once they are set in motion by players". Examples of Game Dynamics are Race to the End, Territorial Acquisition. Dynamics describes the run-time behavior of the mechanics acting on player inputs and each other's outputs over time.

MDA model even mentions the relationship between Aesthetics and Dynamics. It says that Dynamics work to create aesthetic experiences. For example, challenge is created by things like time pressure and opponent play. Fellowship can be encouraged by sharing information across certain members of a session.

Goals

As discussed in the Introduction session, Goals play a major role in Games. Be it a short term quest or a long term aim. The player makes his/her decisions on the goals of the game.

As a designer, you can make the player have a set of goals, or one main goal. Goals keep the player motivated to keep playing the game till the end.

Goals can take form of missions, operations, saving your princess, killing the boss enemy etc.

Theme

Theme or story is the element of a game which is least recognized. Mari Bros. is a game about a plumber on a quest to save a princess; how many even know this. But still, a game needs to have a theme or a back-ground story line on which characters, game-world is created. Theme is the base of visual elements in the game. Sometimes, it even helps in addition of some Game Mechanics. Mechanics that evolve from the theme may be unique of your theme is unique.While you choose a theme for your game, think of something that interests you the most, research about it and try to gamify it. Now that that's out of the way and you understand the concepts, let's move on. Where were we? Ah, the Design phase.

The game design process

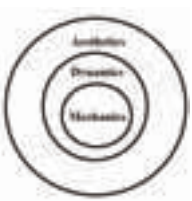
The game design process is divided into three stages:

a. The game idea and concept stage
b. Taking an idea to a formal concept
c. Taking care of game-design constraints

a. The game idea and concept stage

Any game starts with an idea. But where do Designers get ideas from? Do they decide the Game Mechanics first and then work their way to the aesthetics and creating a theme to suit it? Or do they think of a game-world and apply proper mechanics to it? For a Game Designer, it can start anywhere.

Generating a game idea



The MDA framework

Some start with Aesthetics and work their way back to the Dynamics and Mechanics. But majority of the designs start with mechanics which is then followed to aesthetics. According to the MDA framework, one can think of a game as a sphere, with the Mechanics at the core, the Dynamics surrounding them, and the Aesthetics on the surface, each layer growing out of the one inside it. Conversely, a player sees the surface first – the Aesthetics. Some players may be aware of the Mechanics and Dynamics, but aesthetics makes an immediate impression that is most

easily understood. Aesthetics is a feeling in the player which tells whether the game is good or bad, even though the player might not have knowledge of Game Design principles.

As a player keeps playing the game, gradually he might start appreciating the dynamics of the game and now their experiences arise from these dynamics. After understanding the dynamics, the player might understand that they do or don't like the game due to the specific kinds of interactions they are having within the game. If the player spends a lot of time on a game, then they may eventually understand the mechanics involved in the game and understand hot dynamics emerged from these mechanics.

A Game Designer has to understand that he has no direct control over changing the aesthetics and dynamics of the game. If at any point in a game the designer feels that there needs to be a change in a game to "make it more fun", then it can be made only by changing the mechanic.

Approaches that Game Designers follow for idea or concept generation

- **Blue-Sky approach:** This approach allows consider unlimited number of possibilities with no constraints at all. Generally, the Game Designers have constraints on budget, time. But, this approach assumes no constraint except the designer's imagination.
- **Slow-Boil:** In this approach, the designers decide a theme and stuff everything regarding this theme into their minds and wait for an idea to occur. This approach is sarcastically called slow boil, as it might take even years sometimes to come up with an idea with this approach.
- **Mechanic-Driven:** This is the most common approach used. Many famous games like Mario were born by this approach. Mario jumps onto, over and into things and the whole strength of the game lies in this mechanic and makes Mario what it is. Perhaps the largest mechanic-driven games feature the mechanic right in the genre's title: first-person shooters. The mechanic of firing a gun, a wand, or a something evidently never gets old. Even Mario could shoot fireballs.
- **The MDA approach:** Designers can use the MDA approach too. Think of an aesthetic you want to deliver, then think of the corresponding mechanics, dynamics and theme to it. This might be a very strong approach as it it player-centric.
- **IP: Sometimes,** Designers don't start with a new concept at all. Most Designers find themselves developing a sequel to the pre-existing game. Intellectual property is an idea or a collection of ideas that is

owned by someone or something and is the result of creative work. Examples are Duke Nukem, Halo, God of War, Mario, Lara Croft, Spiderman, etc. All of them are legally protected by their owners. According to the Game Market's survey, best-selling game titles are derived from pre-existing IPs. Generally IP games are owned by the developer or the publisher. Big IP games like Halo and World of Warcraft are quiet profitable and have a lot of brand-value in the Game market. Licensed IPs also exist where publishers buy a pre-existing IP for using it in their games.

- **Story:** This is a common approach followed in video games. The story gives an idea of the game world. Sometimes, it might represent the chronological timeline of a person or a company.

b. Taking an idea to a formal concept

Halo

Getting a game idea is the first and a minor step in the Game development process. Many people have ideas; but to bring it to the end of completion there are many more steps to follow and that too for perfection to create a game of quality. A Game Concept is different from an idea. It concentrates on how you intend to entertain someone through your game-play and at a deeper level, why you believe it will be a compelling experience.

To bring your million dollar idea to a concept, you need to sit for hours and formalize whatever is left assumed back in your mind. Some of these would prove helpful:

Making a high-concept statement

Create a high-concept statement. It is generally a 2-3 sentence description of what the game is about. It should describe what feeling it gives to a player and if possible what's new. It should be a "boilerplate" description of your game. This helps in finding publishers and used for promotional and marketing purposes.

For an example, high-concept statement of Bio Shock would look like:

Bio Shock is a horrifying, sophisticated, visually stunning, open-ended experience that will leave players gasping for breath. It's a first person action nightmare that will scare the hell out of people.

Defining player's role

In a game it is very important for the player to understand what he is pretending to be in the game. Playing games often involve playing a role of some sort.

Like in Monopoly, the player plays the role of a tycoon from Russia. Sometimes, the roles in a game are multi-faced: In a sports game, the player's role changes from being an athlete on the field to the coach who is planning the strategy. Hence, it is very important for the designer to define the player's roles clearly as this defines what players actions will be in the game. The designer needs to know what activities he/she is offering to the player.

It is very important to understand that the player has bought your game to play- that is to act; not to hear or see. Hence, interactivity is very important in the game as mentioned earlier in the properties that define games.

Choosing a genre

"Genre is a category of games characterized by a particular set of challenges, regardless of setting of game-world content"
– Ernest.W.Adams, Game Designer.

Genre, by definition refers to the context of the work. Historical fiction, romance fiction, spy fiction, and so on are genres of popular fiction. In games, genre refers to the types of challenges that a game offers. Action games are one type of genre; whatever is the theme of the game-world – old west or space or an army battle ground, they are all still action games. Genre restricts the designer's mind to think in a particular direction which is very good as greater the restrictions of theme and genre, easier for the game designer!

Types of game genres

- **Action games:** An action game is one in which majority of challenges presented are tests of player's physical skills and coordination. Apart from this, puzzle-solving, exploration challenges might be presented as well.
- **Sub-genres:** Shooter games such as *Call of Duty*, combat games such as *God of War*.
- **Strategy games:** Derived from the Greek work "Strategia", which means Generalship; these games expect player to use his decision-making skills to take decisions, keeping in mind the outcome. All Real-Time Strategy games such as *Myth*, *Age of Empires* come under this genre.
- **Role-playing games:** In a role-playing game, players assume the roles of characters in a fictional world. There are several types of role-playing

games – single player, multi-player, massively multiplayer online, real-time RPGs which include MUDs and MUSHes.

- **Sports games:** These are games whose mechanics are based on the rules of the sport on which game is being made. Many famous games such as *NFL* and *EA Cricket* come under this genre.
- **Adventure games:** These are games which test the puzzle-solving skills of the player unlike action games which tests the physical challenge. Puzzle platformers such as *Braid* and *Limbo* fall into this category.
- **Vehicle simulations:** Majority of vehicle simulation games are racing games such as *NFS*. Some of them are vehicle-stunt games.
- **Online Social Games:** Online social game genre has viral game mechanics where a player wants to play, share, and interact with his friends over a social network while playing your game. These have repetitive addictive game-play and use various new concepts such as virtual goods etc.

Choosing a gameplay mode

Propose a proper game-play mode to the player. This has to include the camera model, interaction model and general types of challenges the players will experience in that mode.

The perspective, interaction model, and gameplay is what defines a particular gameplay mode will look like.

Perspective is a means of perceiving the game world and depends upon camera angles. This may include top-down view, side –scrolling view etc in 2D or 3D. Interaction modal includes the way player interacts with world. This may include first person like in first person shooter where the player cannot see his avatar, third person where a player can see the avatar he is controlling or simple puzzles such as *Sudoku* where there is no avatar. Gameplay includes how the player plays the game; it may be a Cooperative, defensive, competitive, stealth mode, attack mode etc.

Thus when you decide the perspective, interaction model and gameplay, you can define the game-play mode of your game. A game can have more than one gameplay mode during its course of play.

Defining target audience

Once the designer is done with the aesthetic experience the designer wants the players to experience, the game-play mode and genre; he/she has to think about who would enjoy that experience. For every game, a target audience has to be decided. This is more of a player-centric design where you test

every design decision against your hypothetical representative player to be sure that the decision helps to entertain your target audience.

Defining the platform

Defining the platform might seem a small issue to the novice game designers; but, it is one of the critical decisions. Defining the platform of the game, defines a lot of things in your game. If this is not taken care in the initial changes, you might end up in changing the mechanics of the game itself. The platform decides the UI, game mechanics feasibility, etc...

You have a chance to choose among a variety of platforms like PC, iOS, Mobile, Console, Web etc. While choosing a platform, you should keep in mind the constraints of technology, Cost of development tools, publishing constraints and what gameplay is best for your game and if the platform supports the gameplay well.

The game-world

As already discussed, the game-world/theme defines a lot of things such as game characters and sometimes even the core mechanics of the game. While you choose how your game world will be, keep in mind the following variables:

- It's dimensions (how large it is, whether it's 2D or 3D)
- Scale (relative size of objects and characters)
- Boundaries (bounded, open-ended)
- Physical Surroundings (buildings, magical items, vehicles)
- How environment changes with time (what happens if time passes)
- Level of detail you want to give (real, cartoon-based)

c. Taking care of game-design constraints

Thinking of a game design is easy, but implementing it takes the hell out of people. A Game Designer faces a lot of constraints in turning his game concept into a game which mainly include the following:

- **Technology constraints:** Difficulties in implementation of the idea, availability of required technology.
- **Budgetary constraints:** How big is your budget? How much you can put in? What is the approximate budget of the game?
- **Development time constraints:** What is the desired time frame of completion? How much time you have before the release?

- **Platform constraints:** What platform are you targeting? What is its reach? What is the target audience for the game? Do they have access to this platform?
- **Ethical constraints:** What is the desired rating for the game? (Although the game won't be rated by the ESRB (Entertainment Software Rating Board) until it's nearly finished production, if at all, publishers have an ideal rating in mind and strive to develop a game to fit that rating.)
- **Publishing constraints:** Does the publisher have any particular genre or category of game in mind? Are there any features that the publisher would like to see in the game? All these constraints play a major role as they might even end up in changing your core-game mechanic.

Writing game design documents

Writing documents is an integral part of a Game Designer's job. Hence, designers produce a series of documents to tell others about their game design. Design documents are very important, as they are used to communicate the ideas properly to artists and programmers with a clarity and certainty.

There are different types of documents produced for different purposes. Let's have an overview at all of them:

High-concept Document

As the name goes, it is not a detailed document from which the complete game can be built. The purpose of a high concept document is to promote and present the game idea before publishers and producers. High-concept doc puts the key ideas of a game on paper so that it can be read in a few minutes and an overview of a game can be understood.

A high-concept document also contains some additional details such as game competition, the unique selling points (USP) that make your game stand out in the marketplace, and possible marketing strategies and related merchandising opportunities.

Game treatment / concept document

This document, very similar to the High-Concept document presents the game in a broad-outline. But this document is presented to someone who is already interested in it and wants to know more details about it. This document is almost like a brochure that sums up the basic idea of the game.

Character-design document
This document is specifically intended to record the design of one character that appears in the game, most often an avatar. This document's primary purpose is to show the character's appearance and it's "moveset" – a list of animations that documents the moves of the character. This document includes plenty of concept art of the character in different poses and with different facial expressions.

World design document
This document contains the back-ground information about all kinds of things that are present in the game world. It is a basis for building all the art and audio of the game world. This document should also document the "feel" of the world, its aesthetic style and emotional tone. The emotions that a designer wants a player to feel through music, images are also indicated in this document.

Flow-board
This is a cross between a flow chart (used by filmmakers to plan a series of shots) and a story board (used by programmers to document an algorithm). Digitally a flow-board can be made using Microsoft Visio; but people find it easier to do it on sheets of paper and stick it on a large wall. Each sheet of paper is used to document one gameplay mode or shell menu. This is done so that everyone in the office can see the structure of the game and revisions can be made easily by adding new sheets and marking up existing ones.

Story and level progression document
Designers make this document to record the large-scale story of the game and the way the levels progress from one to the other. This document is not needed for games which has only one level or is a digital representation of a board-game. This document gives an outline of the player's experience from beginning to end. Here also the designer indicates how the player experiences the story: cut scenes, mission briefings, dialog or other narrative elements.

Technical specification document
This is the 'How' of Game Design. The Designer specifies the architectural vision, technologies to be used in this document. This document

contains the Engineering and Production Details and generally owned by Technical Director. This is generally a huge document and ranges from 50-100 pages which concentrates on the tools to be used, Engines and Middleware, Implementation of different game-play modes, AI implementation. It contains additional details of Networking and server issues, technical risk.

Game design document

The game design document in short is called as the bible of a game, from which the producer preaches the goal, designers champion their ideas and artists and programmers get their instructions. The Lead Designer or the Creative Director is the author of this document with the exception of technical specifications, which is written by the senior programmer or Technical Director. The document contains everything that should happen in the game – number of levels, level-walkthrough, puzzle structure, mission/challenge structure, number of NPCs, project scope, mechanics, detailed game-play modes, objectives, combat, economy, screen management, menus, etc. Look for game design document templates available online.

Game proposal document

This is an extension of the Game concept as this involves gathering feedback and information from other departments, especially the marketing departments. For this evaluation is done in all departments and the feedbacks are properly documented, from which a revised version is made. This document contains the following features:

- **Revised game concept**
- **Market analysis:** This contains the Target market, top performers, feature comparison.
- **Technical analysis:** Experimental features, major development tasks, Risks, alternatives, estimated resources, estimated schedule.
- **Legal analysis:** Any IP issues, Licensing of art assets, Fonts and Sound track etc.
- **Cost and revenue projections:** Resource cost, suggested retail price, revenue projection.

Creating concept art

In the Pre-production stage, concept art of characters and the game-world is done. This is to give an idea and feel about how the game would look.

Having concept art in pre-production stage is useful for game studios that have to present their game to a publisher.

Making your own prototype

Many game studios come up with a prototype in the pre-production stage now-a-days. This is mainly for promotional purposes. Programmers have to understand what the game is about and start coding the base engine or decide to use any open source or commercial game engines which are suitable to the game. Effective Prototypes are made by developers so that desired and important features are known to people whom you want and game mechanics are tested effectively early in pre-production phase.

Generally for early prototype of casual games, programmers use Flash, Game Maker and several other tools discussed below. But before that, you need to:

- **Decide the purpose:** Before you begin to make your game prototype, give yourself some time to figure out what you exactly need from this prototype. You want it for pitching your game to the publisher, or you want it for yourself just to test if the mechanics you designed are fun or not, or you want to prototype to test a new UI design you proposed or any other purpose you can think of.
- **Assemble your prototype kit/tool set:** Selecting the appropriate prototyping tool is not very difficult. You should consider the budget you have to spend on the prototyping tool. In early stages of design, you can use pencil and paper to prototype on paper. This lets you test the mechanics, economy and concept strength of the game.But you often need a digital prototype for video games as a digital prototype allows you to test aspects of your game that paper prototypes lack, like user interface, controls and timing. A programmed prototype is also easy to share with other people and is therefore great for communication and soliciting feedback. You can consider using available prototyping tools as writing one from scratch would be lengthy and will kill the whole purpose of rapid prototyping. Best Solutions tend to fall within the cheap to mid-priced range. The super-expensive solutions are often just not worth it (take more time and money).

Some of best tools for prototypes

FLASHDEVELOP

Flash is without a doubt the best platform for games prototyping. With flash, you can get a games running in a matter of hours. And the learning curve is not bad at all.

Tools you'll need

For Windows users, we recommend installing an awesome piece of free software called FlashDevelop and get running in these easy steps:

1. Download and Install FlashDevelop from *http://www.flashdevelop.org*
2. Download the latest version of the Adobe Flex SDK (located under the Adobe Flex SDK column on *http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+3*
3. Unzip all the Flex files into a location you can remember.
4. Download the Windows Flash Player 10 Projector content debugger from *http://www.adobe.com/support/flashplayer/downloads.html*
5. Download that EXE file and put it in the same folder you placed your Flex SDK.
6. Next we're going to point FlashDevelop to the Flex SDK. Start it up and in the file menu go to: Tools -> Program Settings and choose the AS3 Context tab on the left side of the window. Scroll the window down to the field that says Flex SDK Location. Set this location to where you unzipped the Flex SDK files in the previous step. Then select the Flash Viewer tab on the left side, and set the External Player Path to the location of your Debug Flash Player downloaded in the previous step.
7. Get going to write your new game prototype.

Mac users can use Flash Builder trial *http://www.adobe.com/go/try_flashbuilder*

PROS

- Available Freely, Using FlashDevlop and Flex combo is free, instead of paying up to $700 to buy a license of Adobe Flash.
- For anyone new to creating games, they can use some of best libraries: FLIXEL, FLASHPUNK etc that are free and cover the fundamentals of 2D games, including sprites, tile maps, collisions, and scrolling.
- Support for awesome libraries including TweenLine for "tweening" objects and variables, and Box2D for awesome 2D physics like wheels, ropes and springs.
- Almost everyone with a computer can play it. Upload your finished game on web so you can share it with the rest of the world!
- Using FlashDevelop to code is much easier due to the code predictor similar to Intellisense.
- Code editing in Flash is really bad, so FlashDevelop or Flex itself is better.
- It's Open Source and hence free of cost!

CONS

- In Flex SDK you are almost restricted to using sprites, and does not allow you to code directly on the timeline, which sometimes is very useful in Flash.

GAME MAKER

Game Maker is a Windows and MAC Integrated Development Environment which helps developers to make games without much effort of programming.

PROS:

- Easy-to-learn drag-and-drop actions
- Available for both 2D/3D games
- Cross platform support: Mac, Web(HTML5), Windows(Standard)
- Support for backgrounds, animated graphics, music and sound effects.
- Easy built-in , flexible programming Game-Maker Programming Language(GML)
- Easy scripting
- Internal image and code editor
- Auto-updater to update bug fixes.
- Comes preloaded with a collection of freeware images and sounds to get you started.
- Both a free and commercial version.
- You can sell the games you make and get them published to desktop, mobile and online.

CONS:

- Compilation is rather slow as compared to C++.C etc
- Limited 3D functions

Where can you get this?

You can download this from *http://www.yoyogames.com/*

How much does it cost?

Come with both a free version and a registered version. The registered version, which costs €15 (about $20 or £10), extends the features available in the program, such as the ability to incorporate DLLs, create multiplayer games, take advantage of Direct3D, use particles, and utilize advanced drawing functions.

The free version displays a small Game Maker logo during the loading of the user's game, while the registered version allows the user to replace the Game Maker logo with their logo.

LOVE2D

LÖVE is an awesome framework you can use to make 2D game prototypes in Lua. It's free, open-source, and works on Windows, Mac OS X and Linux.

PROS:

- Utilizes the simplicity and power of scripting language Lua
- Freely available
- Good series of tutorials and demos to help you start with.
- Contains a lot of pre-written code targeted at making games.
- Its API (Application Programming Interface) is very simple.
- Friendly and supportive community.

CONS:

- Lacks a strong UI, drag –drop interface.
- Useful only if you can take effort to learn scripting.

Where can you get this?

- You can download Love Game engine from: *http://love2d.org/*
- You may need to download Lua if you haven't already from: *http://www.lua.org/download.html*

UNITY

Unity is one of the most powerful and accepted 3D game engine at present. You'll get lights, particle systems, collision, input controls, 3D world, cameras, PHYSICS engine and everything else you need to make your first 3D game prototype.

PROS:

- Free version gives you 95 per cent of the paid functionality forever.
- You can make complex 3d games you dream of with ease.
- Cross-Platform (PC/MAC/iPhone/Android/ Xbox/ Playstation/ Wii)
- Excellent documentation
- Powerful rendering, sound, physics engine and networking capabilities.
- 3 scripting languages: JavaScript, C#, and a dialect of Python called Boo
- Great tweaking and debugging support

CONS:

- Proprietary, you are totally dependent of unity and its limit and can't extend it
- Weak Multiplayer support: Anything that requires central server requires you write all network code from scratch.'
- GUI system is quite quirky and slow.
- Difficult to manage source/version control mechanisms with Unity3D.
- Not for 2D games.

Where can you get it?

- Just go to *www.unity3D.com* and download the free version.

TORQUE2D

The world's most powerful and easy-to-use 2D game engine and a product of garage games. Built on the basic framework of the 3D game engine smartly offers many advanced features customized for 2D gameplay.

PROS:

- Allows publishing to Windows, Mac, Xbox 360,Wii, iPhone,
- Intuitive and powerful editor, anyone can jump into game creation with little to no prior knowledge.
- Starter Kits for platfromer, adventure games etc to let you start quick.
- For Torque 2D Pro licensees, clean, fast, robust C++ source code is available so you can optimize your game or extend the engine
- Has integrated networking system that makes implementing event-based multiplayer options easy.
- Powerful sound, graphics and physics engine.
- C++ like scripting language.

CONS:

- It's not free : Pro License requires 99$
- You need extra license for publishing to Xbox360, Wii and iPhone.
- Limited built-in physics (only gravity, friction, etc).
- Unclear documentation.

Where can you get this?

- You can purchase Torque2D pro version from: *http://www.garagegames.com/products/torque-2d*

PYGAME

Pygame is a set of Python modules designed for writing video games. It is built on the excellent SDL library. You can create games using python scripting language and is a handy tool for prototyping.

PROS:

- Uses Python , an easy scripting language
- Truly cross-platform
- Easy to use
- Over 660 projects have been published on the Pygame
- Good tutorials/resources on web
- Free and open-source
- Supports older computer hardware better.
- Lots of open source games available to inspect the source.

CONS:

- Needs understanding of Python scripting before using it.
- Performance-wise does not scale to very large games (which hobby game development rarely is).
- Mostly suited for 2D, although 3D is possible.
- Difficult to distribute as closed source.

Where you can get it?

- You can get the latest build at *http://www.pygame.org/download.shtml*

MICROSOFT XNA

Microsoft XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates video game development and management. It is used widely by independent game developers as it is free, and can be used

PROS:

- Uses .NET languages; managed memory, ease and excellence of the Visual Studio environment, etc.
- Supports both 2D and 3D very well.
- Is proven; look at the Xbox Live Arcade, all of those games are made with XNA.
- Games can be easily run on a networked Xbox

- Development can be done parallel for Xbox, Windows Phone and PC
- Excellent community support through XNA Creators Club and awesome starter kits to get you going
- Opportunity to publish your game on XBLIG (Xbox Live Indie Games).
- Free!

CONS:

- Uses .NET languages; can't use Java, C++, etc, which may be difficult for people who are not comfortable with these
- Windows-only
- Fast upgrades lead to books being out-dated, hinders documentation

Where can you get it?

- You can download it from: *http://www.microsoft.com/download/en/details.aspx?id=39*

MULTI-MEDIA FUSION

The most flexible, powerful, and featured authoring tool available today.

PROS:

- Create prototypes with no programming skills or knowledge required
- Simply start by creating a frame containing any art you want. Then insert and drag objects. Finally, set their behaviors in the easy to use, grid-style Event Editor and you are on your way to great prototype for your game!
- Supports extensions to allow you create even the most complex and fully functional code to handle AI, Dialogues etc.
- Sharing, and even publishing your work in no time
- Flash/SWF Exporter to create content viewable in Adobe® Flash® Player.
- Can be used to develop iOS games using iOS device exporter

CONS:

- Expensive though a demo version is available free.
- Not enough tutorials over the web

Where can you get this?

- You can download demo version/purchase it from the website of Click Team: *http://www.clickteam.com/website/usa/multimedia-fusion-2.html*

Once you decide the prototyping tool you'll be using, make it quick , test the mechanics, UI etc and Iterate quickly by making changes in design and re-implementing in the prototype tool.

Assigning Roles

Towards the end of the pre-production phase where the overall blueprint of the game is ready and the scope of the game is known, the team is assigned different roles. The producer assigns tasks to various programmers, artists etc so that the production phase can run smoothly.

Thus in pre-production phase, a pre-production team, consisting of programmers, designers, artists, writers, producers etc work together for generating an idea, formalizing the game concept, writing the story, putting together comprehensive design document detailing the game's levels, mechanics, gameplay , economy etc and finally prototyping the proposed design for iteration.

II. Production

After the pre-production phase is complete and the game's overall blueprint has been finalized, the development of the game enters the production phase and now a larger team of producers, designers, artists and programmers work towards smooth development of the game. The production phase of the game development is the heaviest in terms of personnel and time resources. This is the time when the team works to its fullest effort. The major tasks during production are creation of various assets, but it needs a significant amount of project management and play-testing/design reviews.

Concept art

The preliminary versions of the concept art that were already made during the pre-production process are now discussed again and after making requisite changes, and getting them authorized, these concept art sketches are frozen and shall now be treated as an official reference for the digital artists. Depending upon the game, the art can be 2D or 3D.

Final concept art

The final and consistent look and feel of the game is decided through sketches. These concept drawings include levels, backgrounds, characters, buildings, objects, vehicles etc. These art sketches are then distributed to the rest of the digital art team (3d artists, texture artists) to help them start

working and help them understand the visual appearance of each of the game asset.

Creating clay models

Clay models are needed when 2D sketches of concept art are not sufficient to depict the geometry of the object/character. This process helps the 3D digital artist understand the concept better. These clay models are easily be scanned into popular 3D software for further detailing. One thing that clay models have superior to organic modeling programs is that you can make cuts clear through the surface. Products like Mudbox, etc. simply can't do that.

Creating blueprints

Understanding 3D concept art may be hard for digital artists due to constraints of perspective, detail etc. Often 3D artists want the concept art to be more detailed. This may include zoom-in parts of some pieces of art to know more detail, object from different perspectives such as side view, front view and behind view. Same as clay models, they can be imported directly into the 3D modeling software to allow the creation of the digital models. Blueprints are made from orthogonal drawings set in top, front and side views.

Tools used

- Pencil and paper for sketches
- Adobe Photoshop, GIMP etc for digital sketches
- Clay, Wax etc for physical 3D models

A blueprint of a character

Level design

Level design is the soul of a game. A player interacts with the game through levels. Level design is both an artistic and technical process. Once the game designer details the game design document in the pre-production, the level designers get accelerated in designing the levels of the game. Level design includes the planning and integration of various 3D objects that form the game level. Various objects that make up a game level such as buildings, rooms, vegetation etc are placed into their corresponding locations. The level designers may need art assets from the art team and some assistance from the programming team to implement level editor and other tools.

Defining level geometry and layout

Every game level needs some structural formations that may consist of terrains, landscapes, platforms and buildings organized in a way to create meaningful gameplay and support game mechanics. It is important to define the level geometry and scale of the objects. This phase requires level designers to lay out the large-scale features of the map, such as hills, cities, rooms, tunnels, etc. where the players, NPC's and opponents can move around. Many game engines include tools for level editing.

Some major geometry types include:

- **BSP (Binary Space Partitioning):** This is used to generate levels that can be rendered quickly while minimizing the number of polygons that need to be redrawn every time the screen refreshes. BSP geometry is used for low detail structural geometry like cylinders, rectangles, stairs, and window frames. For more decorative objects, static mesh geometry is used.
- **Static mesh:** Used for objects that just needs to sit in the level and look pretty. They cannot be animated but can be moved, placed and reskinned in real-time. They are usually not made in the editor but are imported from popular 3D programs.
- **Terrain:** Terrain geometry is based on distance from the camera. Terrain can be set to constantly adjust its poly size based on distance from the camera. So, for example the distant hill may only be a fraction of the polygons that they will be up close. That makes Terrain much faster to use. The down side is that if you ever run at that hill, you'll notice the mountain changing shape as it gains more detail.

Determining level environment

Today video games have highly immersive environments that can leave the player gasp without a breath. Level Designers work together to decide the environmental conditions of the level: Whether it will be day or night, it will rain or flood or fire. The scoring systems, economy and time limit of each level are decided now. They decide how will starting resources be allocated in each level and how will these resources be replenished.

Placing static objects

During this step, game levels are populated with static objects, walls, terrains, buildings, trees, vegetation etc. It's vital for a level designer to under-

stand the gameplay and place these objects at specific locations in the map. During this placement, the designers make sure that the rooms, corridors, buildings etc are easy to navigate when a player plays the level.

Placing actors

In each game level, there are objects that behave as action helper objects or triggers. An actor is anything that has some code attached to it and is used to trigger some event. These objects tell when and where different action sequences, sounds, weapons, healing kits, visual effects etc should appear in the level. They are not seen when the game runs but their visual/audible effects are seen by the player.

For example, an actor called "Player Start" is used to indicate where the player will be once the level begins. The popular level editors come with easy interfaces to easily add actors and assets attached to them.

Placing non-static objects

Here, we specifying non-static parts of a level, such as doors, keys and buttons with associated mechanisms, teleporters, hidden passageways; which the player can interact with.

Placing entities

Now, locations of various entities, such as player units, enemies, monster spawn points, ladders, coins, resource nodes, weapons, save points etc are specified.

Lighting a level

Game level lighting is an important part for level design. Designers convey feelings of surprise, joy, fear, panic etc through lighting. Popular editors like UnrealEd and Unity have specialized lighting engine that allow level designers to render variety of lights in the level environment. Different kinds of light sources can be used and their properties/location can be tweaked to achieve desired results. Lights can be static, dynamic, Omni directional or directed. Light sources can be anything: small or large lamps hanging on walls or from ceilings, the moon or the sun, crystals, lasers and other type of high tech beams, fire, mirrors, magical effects, water surfaces that bounce back light, lava or radioactive slime and so on. Everything is possible as long as there is a noticeable source. Level designers thus have to be careful in lighting game levels.

Placing scripted events

It is the job of level designers to introduce scripted event locations, where certain actions by the player can trigger specified changes in the level environment. Scripted events are any events programmed into a level by the level designer to make any number of things happen. They can be as simple as a player reaching a closet and an enemy suddenly comes out of it or as complex as changing the level exits. Generally they are used to move a story forward as the player to do something specific to keep the story in control. Cutscenes can be considered a scripted event when they are rendered in-engine like fight between two NPCs and player just has to watch it to remain in context with the story. Typical RPG quest lines and missions can also be considered a long chain of scripted events.

More often in single player games, they are used to complement AI by relatively complex actions being scripted. Though it is always suggested to let Ai and physics engine handle such events so as to keep the world dynamic and interactive.

Placing path finding nodes

Though complex path finding solutions are implemented by AI systems discussed later, it is often suggested that level designers can place path-finding nodes that non-player characters take as they walk around to test level navigation and design. Using popular level editors, it is easy to place path finding nodes in a level.

Deciding the level aesthetic

Later stages in level design include adding aesthetic details to the level such as music, textures animation and effects.

Tools used

- **Graphic design software:** Adobe Photoshop, Illustrator, GIMP.
- **Level editors:** Valve's Hammer Editor, Epic's UnrealEd, Leadwerks 3D World Studio, BioWare's Aurora Toolset, id Software's Q3Radiant, Unity 3D and Grome outdoor editor. Multi engine, multi game editors include id Software's GtkRadiant, based on Q3Radiant, and the open source Quark.
- **Games with inbuilt level editors:** *Battlezone 2*, *Cube 2: Sauerbraten* and *Doom 3*, *TimeSplitters*
- **Professional 3D editing software:** 3D Studio Max, Blender, AutoCAD, Lightwave, Maya, Softimage XSI or Grome.
- **Terrain Generator Software:** Terragen, Bryce, Vue, XFrog.

Digital art (2D/3D)

Level designers can never work wonders if the digital art is not up to the mark. In production phase after the concept art is full-proof and accepted by the executives, the 3D/2D artists can start their work. They create the required graphics, illustrations, 3D models, terrains, and landscapes etc. which form the objects seen in the game levels and provide these assets to level designers. These objects can be roughly categorized into static objects, dynamic objects and special effects.

Mesh and polygon modeling

If you ever open a 3D modeling tool, the first thing you can do with it is to create a polygon primitive: cube, sphere, cylinder, etc. These are instant 3D objects. You can understand that these objects if you understand polygons from your basic mathematics background. These polygons are used to create polygon meshes used to create 3d objects and models. The polygon faces usually considered are triangles, squares etc rather than complex polygons for ease of rendering. There are two styles of polygon modeling:

- **Box modeling:** Box modeling is so named because it starts with a box (a cube primitive). The artist then scales the object, moves (translates) vertices and edges, and adds more polygon faces by adding edges, so that they create the 3D shape which they want.
- **Extrusion (edge) modeling:** Little different from pulling and reshaping, it's a way to add more polygons to a mesh. The artist can start with an edge or a polygon. The artist selects either an edge or a polygon face and pulls it to add more polygons to the mesh thus creating a 3D form around its boundary. Usually, the artist creates a 2d shape (polygon) which traces the outline of an object from a photograph or drawing. Then he uses the same image and extrudes the 2D shape into 3D.

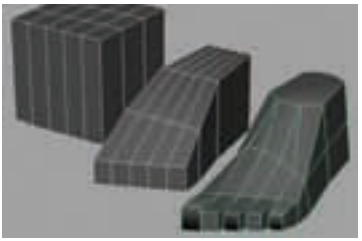
Mirroring

Instead of modeling each side of the object, only one side is modeled. Then its mirror is duplicated and merged into the mesh. Some applications allow you to see the mirror image as you model. This saves a lot of time and ensures a symmetrical model.

Creating edge loops

You can add additional detail in your model by adding edges in strategic places. Some applications allow you to add a ring of edges that loops around

the mesh or some part of the mesh, these rings are called Edge loops. Edge loops are especially practical in organic models which need to be animated. Take a model of a face for example. If the edge loops follow the contours of where the muscles would be then the face will deform much more smoothly compared to when the edges would be place arbitrarily.



Box modeling

Subdividing

You can always add or subtract edges from a polygon mesh. Subdividing is a method of increasing the number of vertices as well as the polygons by adding edges across the mesh. You can simply subdivide your mesh, or you can subdivide and smooth. Smoothing averages out the vertices thus reducing sharp angles in the model. There are different techniques of subdividing Catmull-Clark Subdivision and Doo-Sabine and triangle division etc. Creating characters that come alive on screen is the primary reason that modeling with subdivision surfaces was developed.

Extrusion modeling

Mirroring

Taking care of polygon count

The polygon count is an important aspect of a mesh. The larger the polygon count, the more detail it will have. But it will also take more processing power to manipulate the mesh. This means the more polygons you have, the longer it will take to render. With the right hardware, none of this may be an issue. However, if you are modeling for games, you may very well have limits. In fact, some games have the same model with different levels of detail, keeping low-polygon models when they are far from the camera and vice-versa.

Edge Loops

This controls the overall polygon count of the animated scene that the computer must render in real time.


Subdividing

Deforming/modifying

Deformation is crucial for modeling. It's a great way to mold your object in a way that would be difficult polygon by polygon. Moving a vertex, an edge, or a face is the most basic kind of deforming. Popular 3D programs allow use of brushes to modify the models. They come in different shapes and sizes and can be used to smooth, dodge, inflate and pinch.

Creating a rigid body


Polygon count

There are also surfaces that you'll not want to be deformed during dynamic animations. In those cases, you'll want to make the surface a rigid body. Rigid bodies are great for things such as robots, cars, and wooden swords etc which do not change their course during the game.

Texturing: Texturing/texture mapping is the technique of applying textures/color to the 3D models. Textures are used to give real-world look to 3D models and are created using tools such as Adobe Photoshop, Illustrator etc. One polygon can have more than one texture. This is called multi-texturing and is used to create complex effects.

Preparing 3D model for animators

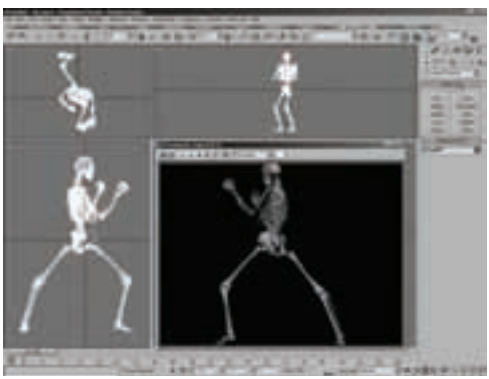
In games, these 3D models you make need to look realistic and thus need to be animated. Thus it is important to prepare your model for animation. Some basic concepts that are required in this process include:

- **Creating soft bodies:** Ever thought how will you model things like a flag or hair etc which react to forces like wind, movement etc. These objects need a different type of modeling than polygonal meshes. They need to be converted into soft-bodies. When you make your model a soft body, you copy the vertices in the mesh and replace the copied vertices with particles. Particles are virtual points that react to simulated forces or even collisions with other objects. Movement of these particles is dynamic and their animation is controlled by forces that act on these particles making it more realistic as compared to manual animation.

- **Morphing:** Morphing is a technique used for animating facial expressions. Here you duplicate the mesh at many places and then reshape the copy as desired. The reshaped copy is called a morph target and will be used for animating the model. Popular 3D programs allow you to use morph modifiers to modify the mesh you are modeling. You can have as many targets as you want, allowing you to morph between lots of different forms of the same mesh.
- **Skeletons**: The skeletons of 3D models are always endoskeletons. They are stiff jointed but they have skeletal rigs as the movers. Every point on the mesh you created is attached to a skeletal rig. These

Texturing

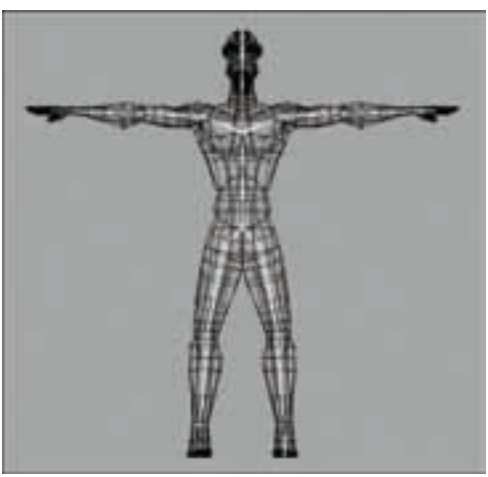
Morphing



Skeleton

points move when the rig moves. The muscles are special deformer lattices being moved by the skeleton and in turn deforming the model to make more realistic movement. Skeletons consist of articulated joints and bones.

- **Adding joints and bones:** Joints are places where your model can be bent or rotated. You can create a joint using Joint tool in any popular 3D modeling software. If you create a joint and then create another joint, a bone and a joint will be created as a child of first joint. This auto-parenting is an important feature of rigs and is crucial. Bones are just connections between joints. Rigging (Creating a skeletal rig): To make a realistic skeletal rig, it is important to understand the anatomy of the character/object you are modeling and then placing joints at appropriate places. For example if you put shoulder joint too close to the neck, the arm will disappear into the body when rotating downwards. If you put it too far away, it will show a gap between the arm and body. Once joints and bones are in place, you decide the bind pose. This pose is a starting position that you can easily go back to, with each joint rotation and transformation set to zero.
- **Skinning:** Once you are done with skeletal rigging and aligning it with your model, you need to connect the rig to your model to move the model surface. This is called binding or skinning. The simplest way to have a rig move a surface is to parent the surface to the rig. There are other methods too such as rigid binding and smooth binding.

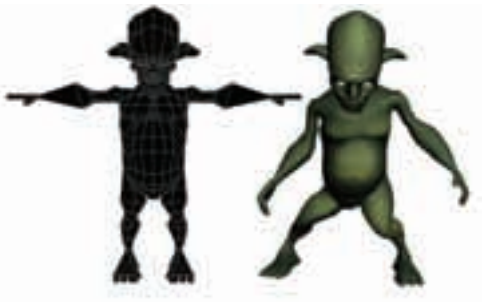


Rigging

Tools used

- 3DS Max and 3DS Max Design
- Blender(Free)
- Lightwave : Student Price: $195
- Maya
- Softimage

- Sculptors such as Mudbox and Zbrush
- Adobe Photoshop/ Illustrator for creating textures



Skinning

Animation and special effects

All movements in a game world are manifestations of some or the other kind of animation. The animator makes the world look real and brings the characters to life. Animations can be effects, animation sequences or programmed actions comprising of graphic programming.

During production phase, the animators animate the 2D designed characters or 3D models to achieve the required effects in the game. The main concepts involved in this process are:

Kinematics

Kinematics is the study of movement of things when forces act on them. In the animation of a rig, it involves figuring out the angles of each joint as it is moved. There are two approaches to this:

Forward Kinematics (FK): With FK, we work directly with the angles of the joints in order to achieve a specific pose. Each joint is moved individually, starting with the base joints towards ones that are lower in the hierarchy. Thus when the parent joints are rotated or moved, all the child joints move as well. You must have an FK rig before you create an IK system.

Inverse Kinematics (IK): It says "If my leg is here, what should be the angle of joints". IK lets you move the last joint in the hierarchy and have all the others follow it.

In order to achieve good, natural movement, you'll use both of them.

Key framing

If still images are sequenced together in quick succession, human mind perceives them as continuous motion and these individual images are called frames. Typical frame rates vary from 24 to 30 frames per second (fps) depending on the format. The computer renders each frame based on

what models, backgrounds, objects, you create and how you control their movement. Key framing is the technique to time the movement of your scene where you define start position, end positions and attributes of the objects such as size, rotation etc.

Creating dope sheets and animation workflow

Once you decide the key poses of the character, you need to time these poses. You can arrange your poses on something like an exposure (X) sheet, also called a dope sheet. On this sheep you write the instructions about each frame. Now take your rigged model and keyframe all the poses. Once you've got these poses keyed in, take a quick preview.

Tweening

Tweening is creation of successive frames of animation between key frames. Here, intermediate frames between 2 images are generated to allow smooth transition. Flash automatically creates these intermediary frames using motion tween or shape tween options.

Onion skinning

While you edit your animation, you can see several previous frames along with the current frame. This technique is called onion skinning and the animator can adjust the animation based on previous frames.

- **Motion capture:** An actor generally wears a skintight suit that has markers on it and performs the actions. The computer can easily pick up the markers as the actor is being shot from different camera angles. The motion of the markers is being recorded as motion data. In the studio, they import this data into popular 3D software to animate the model. Special care needs to be taken to match the rig of the model and the actor for real and smooth animation.
- **Particle effect systems:** Games can contain general effects like fire or smoke that only serve as visual appeal or generally informative purpose. These effects may be running as animated sequences of 3D objects. The effects are usually created with particle effect systems that are automatically repeated throughout the whole game.

Tools used

- Adobe Flash
- Daz Studio, Poser
- 3Ds Max

- Maya
- Sotimage

Audio design

In the production phase, the audio designers design audio assets which the game needs. Since the birth of audio games in the 70s with games like Pong in which game audio was simple blips to fully interactive orchestral scores in today's games have been a long road paved with innovation. Many innovations fueled the success of today's formats including early FM synthesis modules, MIDI, dynamic audio streaming, sampling and DLS, then through to multi-channel audio, dynamic DSPs and interactive music playback.

Onion skinning

Motion Capture

Functions of audio in games

In this section, functions of using game audio in games are mentioned. Game Audio differs from the film audio in some notable things. Depending on genre, platform and on the player's familiarity with a game, some games can function without sound altogether, and as such, can be played with sound muted. Games like GTA: San Andreas gave players freedom to select the kind of music they want to hear by making radio stations available in the game. Game audio for portable platforms are designed with the knowledge that these games are often played in the presence of other people and may require silence. Hence, all the functions that are described below need not be present in every game; but these functions cover all types of game audio.

Commercial functions

In games, it has increasingly become popular to incorporate popular music artists to help sell a game and to help sell the artists. It works on a mutual basis and it is basically for commercial and promotional purposes. There have been cases like in Madden NFL 2003, where the music of some games

have become so popular that, music CDs of these soundtracks were available in stores.

Semiotic functions

Game music helps in conveying emotional meaning, focusing the attention of player, identifying goals and objects. In many games, for example, the lesser enemies all have the same music, and beneficial items like gems or pieces of heart likewise all have the same or similar sounding cues. A crucial semiotic role of sound in games is the preparatory functions that it serves, for instance, to alarm the player to an upcoming event. This can be seen in many horror games like Doom 3 and Silent Hill.

Emotional functions

This is closely related to semiotic functions. This focuses on the functionality of music which affects player's state and creates intimacy. Music induces mood into the player and changes how the player is feeling. For example, in Limbo, a game in which game music is not present for most part of the game, serious music is played when the avatar has to face opponents to create a feeling of anxiety and tension in the player.

Narrative functions

In many cases, audio cues can help to situate the player in the game world, by giving various locales or levels, different musical themes. By listening to the music, the player will be able to identify their whereabouts. Music at the end of boss scenes similarly changes momentarily to let us know we have graduated to a new level, even if physically we remain in the same place in the game. Similarly, audio is often used to locate the player in the storyline, anchoring the player in terms of where/when/what they are.

Non-verbal sound can likewise reveal details about a character whether they are a friend or a foe; for instance, either by their musical accompaniment or by the accent/language/timbre of their voice and, voice-over narrations can let us access a character's thoughts and feelings.

Screen shot of Silent Hill

Immersive functions

Screen shot of Limbo

Audio in games adds realism or creates illusion to immerse the player into the game. This is known as "Suspension of Disbelief". Earlier in films, sounds were used of covering distracting noises of the projector in the silent era of the film. Similarly, this can be seen in Arcade games also. They have more sound effects and percussion to make the player concentrate on the game and cover external distracting sounds from other sources. Now-a-days, in home consoles, game sounds which help to drown out or mask external sound in order to focus on the task at hand.

Kinetic functions

In some games, game sounds are mad in such a way that players respond to the sounds physically. Best example is, Dance Dance Revolution. Some games are expected the player to directly interact and respond with sound. The sound in the case of kinetic games serves as a main motivating factor, arousing the player physically, and is also the part of the game on which the player must focus attention and with which the player interacts.

Aesthetic functions

In the days of the arcades, when there were dozens of arcade machines on a floor, sound was a critical factor in helping to entice a player to a particular game. Hence, sound technology for the arcades grew far more quickly than sound for home consoles, as creators knew the value of having an attractive sound in their game. While the popularity of arcades has faded, sound is still used to create beauty or atmosphere, and can make the difference between a game being soothing or violent. As evidence, sound and music are typically given ratings in reviews and play an important role.

The aesthetics can also give us clues as to what genre of game we can expect. The introductory music to a game that is slow and soft, for instance, usually indicates the game has a slow pace and will take considerable time. Faster music is usually indicative of shorter and more action-orientated games. Certain genres of music adapt well to certain types of games, especially since different types of games have different requirements for inter-

activity in audio. Where sequences of gameplay are short (action shooters, for instance), the music is more likely to be scored, whereas in driving games, with less demand for direct interactivity, licensed music or longer sequences can be used.

Screenshot of Dance Dance Revolution

Game music

Music in games started in the 70s after implementation in games like "Pong" and "Spac Invaders". In those times, vast majority of commercial music was being recorded and delivered on phonograph records and cassette tapes. As these media were expensive and impractical to implement, developers initially turned to digital music generation.

The first leap-frog in the quality and presence of game music in home-consoles occurred with the NES. And from then on, till date, the game music evolution has witnessed a steep increasing curve.

Tools used in creating Game music

There are various tools and middle-wares, with the help of which one can create decent music for games. Some of them are cross-platform and some are platform specific.

i) Cross-platform

- **FMOD:** This tool has been used to create music for famous games like L.A.Noire, Crysis 2, Mafia 2, Starcraft II, Far Cry, Batman: Arkham Asylum. It supports multiple-platforms which include PC, MAC, Linux, Game Consoles and Portable devices. It is available for free which will allow you to use it for non-commercial purposes. To use the tool for commercial purposes, Commercial Licensing has to be done. This tool supports multiple formats which include: AIFF, ASX, DLS, FLAC, FSB, IT, M3U, MIDI, MP2, Ogg Vorbis, PLS, S3M, VAG (PS2/PS3 format), WAV, WAX, WMA, XM, XMA (only on Xbox 360), as well as raw audio data. FMOD is integrated with some famous Game Engines which include

Unity, Unreal Engine 3, CryEngine, Torque Game Engine, and Big World Technology. You can find more information about FMOD here: *http://www.fmod.org/wiki/index.php5?title=Main_Page*

- **Miles Sound System:** It was formerly known as the Audio Interface library and was used in over 500o games since its first release in 1991. Some of the famous ones include: Metal of Honor Allied, Guild Wars, Turrican 3D, Z.A.R., Black Ops, Robot Wars, Star Trek, Age of Mythology. It supports multiple-platforms which include: Windows, Nintendo Wii, Nintendo 3DS, Microsoft Xbox 360, PSP, PS2/PS3, MAC OS, iPhone, Linux. Miles boasts of having wide variety of features, which include: High-Level Sound Authoring, High-Level Auditioning, Digital Audio Support, Blink Audio Support, MP3 support, DSP Digital Filter Support, Platform Support and Source Code details.
- **Audio Kinetic's Wwise:** Wave Works interactive Sound Engine (Wwise) supports multiple-platforms which include PC, Xbox 360, PS3 and Wii. Some famous games which used Wwise are: Mass Effect2, Assassin's Creed 2, Assassin's Creed: Brother Hood, BioShock Infinite, Halo Wars. It consists of some interesting features which allows sound Designers to Mix-in Real-Time, Define Game Stats, Manage Sound Integration, apply audio and plug-in effects.
- **BASS:** It is an audio library which supports Windows, MAC and Linux. It supports various formats which include MP3, MP2, MP1, OGG, WAV, AIFF, XM, IT, S3M. It provides C/C++, Delphi, Visual Basic and MASM APIs.
- **Other Audio Libraries:** Other audio libraries like Open AL, Audiere, Renderware Audio provide audio APIs for games which support multiple-platforms.

ii) Platform-specific tools

Interface of FMOD

Xbox development: Xbox provides a regular DirectX SDK with Xbox audio creation tool for PC called XACT. It runs on PC and predicted to be the Next generation audio tool. It supports WAV, AIFF, XMA formats. The XACT API allows features which include in memory and streaming support, audio event notification. Microsoft's documentation on XACT can be found here: *http://msdn.microsoft.com/en-us/library/ee415932%28v=vs.85%29.aspx*

Programming the game

Programming is one of the most integral parts of the game development. Since, the program code binds the whole game together, the task of programming continues from beginning of the project to the end. Programming can be done in various levels of detail – source code or script- but no matter what the approach; the aim of the software is to create the structure and dynamics of the game.

Basics for newbies

We have already gone through various tool which help programmers create prototypes. But, in the production phase, the complete game has to be made. The aim of the teams of programmers working on the game is to make a complete bug-free application which typically comprises of the following:

Initialisation

This is the first few codes of a game program. This is the code which is responsible for a window or interface that is created which will contain the main screen of the game. After that, graphics sound and network interfaces are initialized. The initialization code contains code for menu and customisation options. This is the part of the code where global values are initialized, memories are allocated, and resources are loaded.

Interface of Bass

Main game loop

The start of the game loop is where all the action begins and where the application starts to prepare all the necessary things that go into the presentation of the next frame. After the next frame is presented, the game loop continues until the player decides to exit from the game. What happens throughout the game loop is determined by the current game state. A game state could be anything such as menu navigation, a level, or a cut scene. Generally, game data is updated in the following order:

- Player related data updates which include player's input according to

which player's state according to the game-world is updated, taking into consideration of the world-restrictions.

- World related data updates which include passive elements which are static and logic-based elements which are dynamic.
- AI-based updates which include updating goals and states of NPCs, update the decision engine and the game world according to the change in player's state.

Approaches of updating game-Loop

There are three approaches that are followed to update the game loop:

i) To design update/render the process in a single loop

Advantages

- Both routines are given equal importance.
- Logic and presentation are properly coupled.

Disadvantages

- Variation in complexity of one of the routines will influence the other.
- There is no control over how often a routine is updated.

ii) To Design update process using two threads with decoupled frequency

Advantages

- Both update and render loops at the same time.

Disadvantages

- Not all machines are that good at handling threads. As a single CPU has to handle it, precise timing problems can occur. Synchronization issues, as two threads accessing same data

iii) To design an update/render single threaded decoupled process

Advantages

- It has better control than thread and has simpler programming to do.
- No need of sharing and synchronizing.

Disadvantages

- It assumes that the timer takes o sec. to complete.
- It does not handle Alt-Tab scenario.
- There is no nesting with
- different frequencies.

Player input

A game needs Input and needs to be coded by a programmer. The player input section of the game program is where the player input is retrieved and processed or buffered to be used in the game logic and AI section of the game.

Game logic and AI

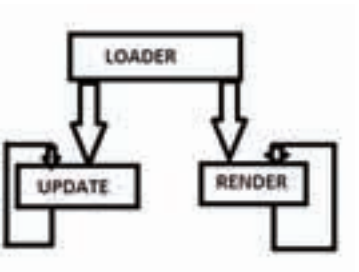
This is the section where majority of the game programmers work on. This is where all of the creativity of a game developer is put in. This section of code contains everything from plot to concept and gameplay will be written code by code.

Rendering of the scene

After the game logic is finished, the output is brought to the rendering section. Every object in a game has its own corresponding set of data that represents its coordinates, orientation, movement, animation, and its interaction in the gameplay. This data will then determine how a polygon, or an image surface, or the effects associated with the object will be displayed on the screen.

Game exit

This set of codes is executed whenever a player decides to exit the game. Before a player can completely exit from the game, all the resources are released, memory is de-allocated, and the system is cleared from any temporary data.



Decoupled flow diagram

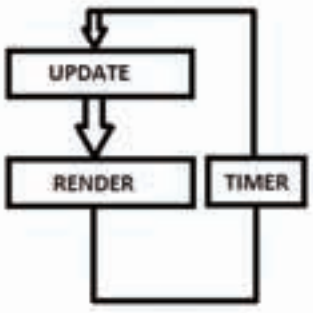
Game engine programming (Writing the core of your game)

The pre-requisite to program Game Engines is strong knowledge of C++ and Object-Oriented Programming. C++ is an industry standard that is followed, on which many games are programmed. Java is rarely used for console-based games; J2ME is used for mobile-based gaming platforms though. C# is used in combination with XNA to produce Xbox games.

What is a game engine?

The Game Engine is a library of core functions used in the game, related to

graphics, input, networking and other systems. The library of core functions that are created by a set of programmers are used by other programmers and use these functions to code for other parts of the game. Different types of genres require different types of requirements according to which Game Engines are made.



3rd Loop flow diagram

Technology requirements of different genres

- **First-Person Shooter Games**: Require efficient rendering, physics-based animation and AI of NPCs.
- **Platfromer and Third-Person Shooter games:** Requires dynamic world, high fidelity animation of main character, camera collision system.
- **Fighting/Action games:** Requires animation database, accurate user input, character animation.
- **Racing games:** Requires level of detail, Rendering, Rigid body physics and deformations.
- **Real-Time Strategy (RTS):** Requires good AI, proper programming for evolving environment.
- **Massively Multiplayer Online Games (MMOG):** Requires data optimization, memory and account management, VoIP service, intensive use of network.

Game engine architecture

There are three layers: Game Engine API, Hardware Abstraction Layer, and Hardware Layer. Hardware Layer consists of sound card, Graphics card and physics card. In the Game Engine API, engines for specific tasks are made:

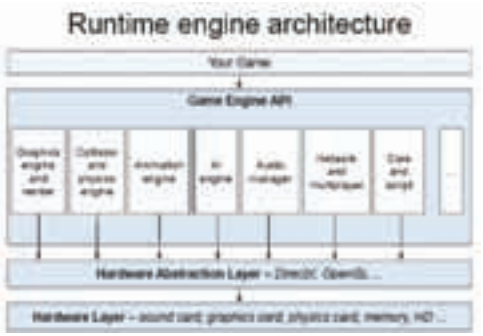
Graphics engine

- Handles the Graphic rendering of the game.
- Used to model data
- SDKs like Microsoft DirectX or OGRE can be used to integrate with the game engine.
- The Graphic Engine is built on the top of human interface library.

- The Graphic Engine takes care of low-level optimized scene graph exploaration and rendering, visual effects and front end HUD, menus, GUI.
- Handle shadowing and Lighting in the Game world.

Collision and Physics engine

- Handles simulations of the game world like physical Behavior, Collisions, Terrain changes, Explosions, Object breaking and destruction.
- Physics Engine takes care of physics-based animations.



Game engine architecture

- Some famous physics Engine SDKs include – Havok, ODE (Open Dynamic Engine), Tokamak, PhysX.

Animation engine

- It is partially linked to physics Engine.
- It handles off-line motion capturing and retargeting, motion editing and annotation.
- It handles real-time sprite/texture animation, vertex animation, rigid body and skeletal motion.
- Some of the famous Animation Engines are: Granny (majorly used in games), Havok Animation, Edge (for PS3)

AI engine

- Handles the behavior and Interaction of all the NPCs in the game.
- Handles all the spatial displacement like obstacle avoidance, path planning.
- Creates the base code architecture for scripting Strategies in the game world like Hiding, Attack/Defense.
- Handles the Decision making of NPCs in the game.
- Some famous AI game Engines include AI Implant by Presagis, Kynapse by Autodesk, DirectIA by MASA, SimBionic.

Architecture of an AI engine

- The AI engine follows the Architecture shown in the graph:
- Perception: Perception of the AI Engine is programmed to sense vision, dialogues and other inputs from the environment in the game world. This contains all the libraries which will make an NPC perceive what is happening in the game world.

Dynamic shadowing techniques used in Cry engine

- Decision: This contains libraries which will help the AI programmer to program the core AI which include rules/learning process.
- Action: A set of libraries which defines the various types of actions, an NPC can perform.

Audio engine

- Some Game Engine APIs come with Audio Manage in-built. Example XACT for Xbox.
- This will take care of the sound design and game music.
- To get more information on Game sound and Music, go to Audio Design section of this chapter.

Animation engine granny

Networking engine

- This part of the Game Engine API, allows multi-players to play together with a shared virtual world.
- Data can be transferred between players/computers.

Types of multi-player modes

Single-screen multiplayer

- A multi-player mode in which there's a single screen and all the players are connected to it by game-pads
- Used in the case of Action/Adventure games.
- Common in case of consoles.
- Single camera in operation

Split-screen multiplayer

- Allow multi-player on a single screen by connecting multiple gamepads.
- In this, screen is split into different sections.
- Multiple HID and cameras.
- Used in the case of First-person shooter games, where each player peeks into the other player's screen. This is considered to add more fun to the game.

Networked multi-player

- Multiple computers networked.
- Used in the case of Real-Time Strategy, FPS games.
- Generally played on LAN, local servers.

An example of a LAN party

Massively multiplayer

- Supports thousands of players at a time and are played on the Internet.
- Hosted by a central server.
- A persistent world exists where people interact with each other.
- Best example is World of Warcraft.
- Memory management is critical in these games.

Scripting in games

These are computer languages which help to reduce the edit-compile-link-run process. Some scripting languages extent languages like C or C++ and some are stand-alone languages like python, Lua, AngelScript, Perl. The scripting languages compile on a virtual machine (VM), which is a separate piece of software that isolates the main application from the computer.

Here's why you need to add a scripting language to game engines:

- Scripting gives you the ability to code game's major functionality without having to compile the Game Engine.
- Those functionalities include Artificial Intelligence (AI), User Interface (UI), game events, save and load game functionality.
- Advantage to modders to change the look and feel of the game.
- Scripting allows people of a game company of various disciplines to work on their piece of code, without changing anything in the Game Engine.
- For example, say that a class for an object is coded in the game engine that can be placed in the game world. If a Level-Designer wants to place that object at a particular point in the game-world, hard-coding the coordinates of the world space and tweak the Game Engine will not only be tedious but also waste of time. Scripting in games would avoid this.

A split- screen game

Disadvantages of embedding scripting a game engine

- It is relatively slow as it runs on Virtual Machine.
- Scripts can't be used to create entirely new visual effects, because of lack of speed.
- People who are working on the Scripting Language have to learn the Language properly.

Embedding the script engine

- The Script Engine needs to be bound to the Game Engine so that a two-way communication can take place.
- The Script Engine code is embedded in the Game Engine code by linking explicitly or implicitly to a DLL or static .LIB library. This is not the only way to do, but it does represent the scenario in Angel Script and Lua embedding scenario.
- Hence, by this embedded script, the programmer can call the functions of

the Game Engine to access local, class and global game engine variables and class objects as well. As it runs on a Virtual Environment, there's no fear of Game Engine crashing.

Tools used

Famous Commercial Game Engines

- Id Software's DOOM, Quake Engine - FPS engine
- Epic games' Unreal Engine
- Microsoft's XNA framework – development platform for Xbox 360.
- Valve's source Engine – used in half-life and counter-strike.
- Torque Game Engine.
- Unity 3D

Open source game engines

- Panda 3D
- Yake (OGRE 3D) – Rendering Engine
- Blender Engine
- Irrlicht

Existing game engine versus writing your own

- Less development time required.
- Less testing and debugging.
- Many features directly available.
- Better focus on the game design.

Why to create a new engine

- No control over the implementation of features
- Adding features not yet in the game engine might be time consuming
- Dependent on other licensing scheme for release
- Other libraries/toolkits linked with the game engine (physics, AI...)
- A new Engine can be customized to your own needs.

Play testing

Play testing is set of techniques and methods that are used to collect relevant data and feedback about a player's experience with the game. Game Design play tests are used to analyze game usability; it's UI, balancing, level difficulty, mechanics, memory load, multi-player options etc. Play testing is different and harder than playing. There are several techniques

and methods for effectively play-testing your game but all of these rely on the following basic concepts:

- Let the play testers play the game without, little or no interference and find other ways of collecting feedback such as observing, recording play tests etc.
- Never justify the design flaw or technical bug.
- Collect play-test data objectively and honestly.
- Accept feedback and make relevant changes.

Engine versus scripting

What belongs to the Engine and what to Scripting?

	ENGINE	SCRIPTING
GRAPHICS	Rendering Shadows/Lighting. Occlusion culling	Time- of- day Add/Remove lights. Loading/moving objects
PHYSICS	Dynamics Collision Response Ray-Casting	Object mass/friction. Collision events. Ray cast events.
ARTIFICAL INTELLIGENCE	Path finding Fuzzy controllers Planning	Path selection. Decision making. Goals/objectives.

In contrast to software testing which is more focused on measuring outcomes, game play testing is done to evaluate the overall user experience though there are some play tests done to evaluate frame rate, memory requirements, synchronization etc.

Some of the major tasks during play testing include:

- **Identifying and prioritising bugs:** Once the play testers play your game, the bugs need to be identified. As game designer you should evaluate play tests for variables such as fun, playability, speed, gameplay etc and the problems need to be identified and design/technical solutions to these problems need to be thought about. Once you have a list of bugs, it is the task of the game designer to prioritize these bugs as which of them have to be fixed first. The prioritizing of bugs to be fixed is important as some bugs are more crucial to the game than the others.

- **Reporting:** After the evaluation of the play test, the designer reports the design changes to the art, animation and programming team to follow and iterate accordingly. It is important to cross check bugs between various versions and different play testers.

Types of testing needed for video games

Video games need some special types of testing which usual software may/may not need. Following are the types of testing required for games: Functionality Testing: This testing requires testing of general game functions and gameplay and does not need any technical expertise. Usually done by external play testers, this type of testing is done during earlier phases of the game development concentrating upon revising game mechanics and gameplay.

- **Compliance testing:** This type of testing is crucial for console development as console platforms have strict guidelines for technical requirements. Microsoft, Sony, Nintendo all have some sort of technical requirement checklist to comply to. Some other tests during this testing include formatting of standard error messages, legal authentication of copyrighted material (if used), testing for objectionable content and memory card data handling.
- **Compatibility testing:** Here the game is tested on a set of platforms listed by the publisher for any compatibility issues regarding hardware.
- **Localisation testing:** If game is to be released worldwide or at more than one country, it is important to test if the game sounds, UI, text, dialogues, symbols etc are properly changed and localized.
- **Soak testing:** Since game is a form of interactive entertainment, this form of testing becomes very important for games. Since a game can be left at idle, exit, paused state by the player, the game has to be tested for memory leaks and rounding errors. This technique is called soak testing.
- **Regression testing:** This is used to test if fixing of one bug lead to another bug or not.
- **Load testing:** In case of MMORPGS and other online games like World of Warcraft, this type of testing is important to test load on the server.
- **Multi-player testing:** This type of testing is essential for games that support multi-player mode. The game is tested for different types for network configurations, synchronization of game variables etc.

Applying testing techniques

There are several testing techniques which may be used as per the requirement of the game and designers. Trying to test every possible combination of game event, function, and configuration is neither practical nor economical. One popular technique is combinatorial testing.

- **Combinatorial Testing:** Here you choose parameters like game events, gameplay options, hardware configurations, character choices etc and test them pair wise. Here combinatorial tables are constructed with parameters to be tested. For example whether a player chooses to play as Mario or Luigi , chooses a black or a white suit etc.
- **Test Flow Diagrams:** One other technique is to use test flow diagrams. Test flow diagrams are a formal approach to testing with a high degree of reusability in sequels, similar game events etc. They are easy to analyze and understand because of their graphical nature and it is easy to take feedback from developers.
- **Test Trees:** Test trees are a set of tree graphs made to indicate the testing process. They are used to document and understand hierarchical association of test cases and game features and functions. They can be mad using pen and paper or using some software. Every time a new build is sent for testing, these test trees are updated.

Bug fixing and verification

After the developers fix the bug as directed by the game designer or technical director, this bug fix has to be verified by the testers. Once verified, the bug is deleted from the bug list. However some bugs need not be fixed as they are minor and thus are waived by game designers and producers.

- **Quality Assurance:** This process takes place through-out the Production phase. Quality Assurance is the systematic monitoring and evaluation of various aspects of the game. It focuses on the quality of the game and aims at assuring best possible quality. The most important method for successful QA is a formal review system that covers all aspects of Design and Production. The following are the testing methods followed in QA:

Final testing

This follows the traditional software testing mechanism of Software Engineering. Here, the cod of the game is tested by conducting a series of Formal and Informal tests, which are already discussed.

Alpha testing

It is the operational testing of a game, tested by potential users/customers or an independent test team at the developer's site. For this, professional testers are hired; bugs are checked, features to be added are listed. This phase makes the game ready for Beta Testing.

Beta testing

It is an external user-acceptance testing. Versions of software known as beta versions are released to a limited audience outside of the team. This testing is conducted to get a feed-back of the players. It has to be kept in mind that Beta Testing can be conducted only when the game contains no bugs. Beta Testing means all the known bugs have been corrected and there is no system or feature to be added in the game.

III. Post-Production

Once the game is complete, it enters the post-production phase. This phase includes extensive testing, review, publishing, marketing and finally, distribution.

Testing and review

Testers play your game repeatedly in continuation to play testing done during production phase. The major and minor bug fixes done during production phase are fixed and a beta version of the game is released (often open to public). This beta version is excessively tested and reviewed until in-house team of developers, testers etc are satisfied and even the public beta receives a good feedback.

Obtaining ESRB rating

Parallel to the testing, a copy of the game is sent to Entertainment Software Rating Board (ESRB) to be reviewed and a rating is given to the game based on its content.

Master gold-release

This is the final version of the game which is ready for printing and publishing. In this phase, the incremental portions of the game are frozen. Hence, no more code, art or design to be added.

Publishing

Games like movies need publishers. For independent game developers,

students and hobbyists developing casual games, publishing on AppStore is one of the best and easiest options as Apple is very cooperative and if your game is good, it will find a place for itself. XBLIG (Xbox Live Indie Games) is another way to digitally publish your game. Though XBLIG sales are not that high, it's a great option for students and hobbyists to get people play their game.

Example Test flow diagram

If you're making a PC game, you can get it published through STEAM or any other similar digital publishers. If you are making a Facebook game, you can publish it via Facebook etc. You can also use game portals that offer various games and huge amount of players. Their fees typically range from 50-65 per cent net sales. Some popular portals include Arcade Town, Big Fish Games, Grab, MiniClip, MSN, Pogo, Real Arcade, Reflexive Arcade, and Yahoo Games.

If you are making a hard core game, you need to find some publisher who should like your game and will be ready to publish it. One way is to self-publish in the net. This is a very common way for game developers to start selling their game as it just needs a web host and you need to handle all advertising and marketing yourself. There are numerous ways to publish your game depending upon its type, genre, target audience and platform. There's no one best way, it's on you to figure out the right way for your game.

Localisation

Games very similar to Hollywood movies release world-wide. For suiting various market areas and cultures, the content of the game may be changed. This process is known as "Localisation". This plays an important role, as in some countries the game which you are about to release may not be accepted due to cultural or religious differ-

ences. Hence, the game has to be substantially modified in order to be acceptable for release in that region. Localisation requires a lot of initial planning as the game's theme, settings or features might have to be changed, which cannot be done over-night. Hence, this has to be kept in mind by the Designer and design the game with respect to his target audience.

Porting

With so many platforms on which players can play a game, it is important to port your game into different platforms. For example, Angry Birds was made on iPhone first and was one of the biggest hits ever. Then, it was ported into other mobile platforms like Android, Windows Phone. This was done because, Android also had a huge market potential of Casual gamers. Because of porting into other platforms, Angry Birds earned almost double to what they have earned on iPhone alone.

Marketing

In today's competitive world, marketing your game is one of the most important tasks you need to focus on. After all, you made the game to sell it and make some money. Companies spend more for marketing the game than the development cost. A blockbuster game like Grand Theft Auto: Vice City can cost between $3 million – $5 million to develop, with an addition $10 million for promotion and marketing. With hell out of money at stake, publishes make sure that their game is well marketed, hyped and promoted. Though major marketing happens in the post-production phase, some sense of marketing begins early in the development process. The development teams, producers etc visit game festivals, trade shows like E3 and conferences like Game Developers Conference around the world to promote their latest games. Demonstration versions, or "demos," are sometimes made available for download from the Internet to invoke interest in buyers of final game. Trailers play a crucial role in speaking about your game and should be made with a lot of care and effort. All of this is done so that buyers will be willing to spend their hard earned money to be immersed in the imaginary world that that game developers worked so hard to bring to life and make it hit the floors.

Distribution

Once the game is released, the game needs to be distributed and this is one

of the major responsibilities of the publisher. The success of a game depends on the scope of its distribution. Publishers offer a wide range of distribution channels and possibilities for promoting your game using their expertise and reach. You can distribute your game in physical form in game CDs or in digital forms through internet. Digital distributors such as STEAM, On Live, and Netflix etc have maintained a reputation in digital distribution. You can reach mass markets by distributing your games through Facebook. Whatever medium of distribution you choose, it is important to understand the pros and cons effectively.

IV. The aftermath

When the game has been released, most of the developers need their time to recover from the hectic crunch time. However, there is one final phase that needs to be tackled before the developer is completely over the project. Post-release phase is where you plan what more you can do to increase the lifespan of your game and to keep your players stay with your game.

Planning updates

Developers usually release updates to their game to make more money out of the game and to keep the players interested in the game. These updates can be some added achievements, a new festive level, a free add-on or any other kind of upgrade. Today it is very common to release updates to your game on casual game platforms like iPhone as it keeps your target audience plugged-in to your game and increases your revenue potential.

Planning sequels

While players keep moaning about the existence of sequels quoting that there's no innovation, repeated mechanics etc. the reason why publishers and developers develop, ship and sell sequels to successful or moderately successful games is that "Players do buy sequels no matter what they say". Publishers and developers publish sequels to mitigate the risk involved in the creation of a new IP/title. There are several reasons why sequels are developed:

- Proven technology
- Opportunity to make the game better and incorporate feedback
- Use of potential unused assets of the previous title
- Promised buyers

There can be different types of sequels including expansion packs, mods, clones, actual sequels and yearly releases.

Thus game development is a rigorous process involving huge time and effort. It is important to schedule you your game and draft a production plan according to phases explained in this Fast Track.

TIPS – WHAT WORKS AND DOESN'T

Not everyone will love your game, but here are some tips and tricks that can help you gain visibility and recognition for your efforts

As a developer it's important to make the decision about what platform you're targeting your game at. With cheaper handsets available, developing for mobile devices is one of the most common choices among indie developers in India. A game targeted at the Indian audience will be better off if made for Android devices rather than iOS devices, as more people own an Android device in the country. There's also a huge diversity of devices, counting Android version numbers, screen sizes, resolutions and aspect ratios. You can choose iOS as a platform, over Android if you find this cumbersome. When you're developing a game for a platform, it should run on all devices supported by thatplatform. A game made for iOS should run on the iPhone 3G, 4G and iPhone 4S. You need all these devices to be able to testing the game before you release it.

A game designed with touch controls may need a redesign of controls when ported to a non-touch device. A game ported from PC to iPhone may need a lot of graphic and texture optimisation due to lesser available memory.

There's an industry standard for mobile computer graphics, called OpenGL ES (OpenGL for Embedded Systems), which is not supported by Windows Phone, as it uses Microsoft's own XNA for these purposes. So if you're learning Microsoft XNA, you can build for both Windows Phone and Xbox 360. However, some of you may choose to learn OpenGL ES instead, and focus your attention on the larger market share.

Know your audience

Game genre plays an important role in defining your target audience to an extent. Action game lovers will automatically become your target audience if your game falls in the Action genre. Casual games are made to be played by casual gamers, and are thus targeted at smart phones, tablets, etc. Hard-core gamers, on the other hand, love playing FPS or RPGs on their consoles and not *Angry Birds* or *Fruit Ninja*. If Facebook users are your audience, design your game using the network's viral features – your game will need to ensure that people can interact with friends within the game.

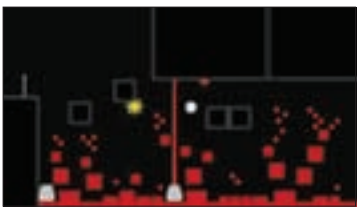
You can't make one game that the world will love as a whole, so don't try to achieve that. If you try and create a game that will capture both casual and hard-core gamers, you will probably make a game that neither group likes.

The next important decision variable is age. Decide whether your game is targeted at kids, teens or adults – this will influence factors such as colors, content, etc., of your game.

Keeping social factors in mind is also important these days. For example the game *Hanuman: Boy Warrior* was not received well by audiences because people didn't take to the idea of manipulating a holy figure with a joystick.

Importance of sound and music

Music and sound effects form an important part of games and should never be neglected while developing a game. Games such as *Braid* and *Contre Jour* won many awards and recognition due to their melodious music. Music is important for building atmosphere within games and evoking certain kinds of emotions in gamers. As a designer, it's good to consider the kind of feel you want your players to experience. *Limbo*, a puzzle game has no background music, but sudden, atmospheric and horror sounds lend a scary feel to the game. The sound of stretching the catapult in *Angry Birds* gives you the feeling of actually stretching the catapult. Timing and style of music are two crucial ingredients of sound design in games. *Sound Shapes* is an upcoming PlayStation Vita game designed by Jonathan Mak which is entirely focused on sounds and music.



Screenshot from the game Sound Shapes

Being innovative

Any kind of Innovation whether in art, technology, controls, game concept or music is appreciated in games. When you're designing the game, think of ways in which you can make it stand out in comparison to other games. *Tiny Wings*, an iPhone game developed by only one developer became a hit because of its innovation in procedural graphic generation. *Jetpack Joyride*, another iPhone game was well received due to its highly addictive nature. There are numerous examples of how innovative games have found a place for themselves in the market. What's important is understanding that the game should always serve its vital function – to entertain.

Procedural graphics of Tiny Wings

Appendix

GAME-DESIGN RESOURCES:

Books:
- Challenges for Game Designers by Brenda Brathwaite and Ian Schreiber
- Principles of Game Design by Andrew Rollings and Ernest Adams
- The Art of Game Design: A Book of Lenses by Jesse Schell
- Rules of Play : Game Design Fundamentals by Katie Salen and Eric Zimmerman
- Game Design Workshop : A Play centric Approach to Creating Innovative Games by Tracy Fullerton

Online Resources:
- http://gamedesignconcepts.wordpress.com/ – Online game design class
- http://gamebalanceconcepts.wordpress.com/ – Online game balance concepts class
- http://www.alanemrich.com/ – Good compilation of game design concepts
- http://www.sloperama.com/advice.html – Tom Sloper's advices on Game Industry

GAME-ART RESOURCES:
Books:
- 1000 Game Heroes by David Choquet
- The Art of Game Worlds by Dave Morris and Leo Hartas
- Half-Life 2: Raising the Bar by David Hodgson
- The Art of God of War III (The Art of the Game) by Daniel P. Wade
- Game development essentials: video game art by Todd Gantzler
- 3D Game Textures, Second Edition: Create Professional Game Art Using Photoshop by Luke Ahearn

Online Resources:
- http://www.game-artist.net
- http://www.deviantart.com/

GAME-PROGRAMING RESOURCES:
Books:
- Game Development Essentials: An Introduction by Jeannie Novak
- Real-Time Rendering, Third Edition by Tomas Akenine-Moller, Eric Haines, Naty Hoffman
- Mastering Unreal Technology, Volume I: Introduction to Level Design with Unreal Engine 3 by Jason Busby, Zak Parrish, Jeff Wilson
- Beginning C++ Through Game Programming by Michael Dawson
- Game Engine Architecture by Jason Gregory, Jeff Lander, Matt Whiting
- Learning XNA 4.0: Game Development for the PC, Xbox 360, and Windows Phone 7 by Aaron Reed
- Programming Game AI by Example by Mat Buckland

Online Resources:
- http://www.gpwiki.org/forums/ – Wiki and forum of Game Programmers
- http://www.dedge.com/online-game-development.php

GAME PRODUCTION RESOURCES:
Books:
- Game Production Handbook by Heather M Chandler
- Game Development and Production by Erik Bethke

Online Resources:
- http://www.gameproducer.net/

GENERAL ONLINE RESOURCES:
- http://www.gamasutra.com/
- http://www.gamedev.net/
- http://www.gamecareerguide.com/
- http://www.igda.org/
- http://www.gdmag.com/homepage.htm